TUGAS AKHIR – TI 141501

IMPLEMENTASI ALGORITMA *HYBRID CROSS ENTROPY – GENETIC ALGORITHM* UNTUK MENYELESAIKAN *SINGLE STAGE CAPACITATED WAREHOUSE LOCATION PROBLEM* (STUDI KASUS : PT. PETROKIMIA GRESIK)

FATMAH MUNIF LAHDJI
NRP 2512 100 023

Dosen Pembimbing
Prof. Ir. Budi Santosa M.S. Ph.D

JURUSAN TEKNIK INDUSTRI
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – TI 141501

IMPLEMENTATION OF HYBRID CROSS ENTROPY – GENETIC ALGORITHM FOR SOLVING SINGLE STAGE CAPACITATED WAREHOUSE LOCATION PROBLEM (CASE STUDY : PT. PETROKIMIA GRESIK)

FATMAH MUNIF LAHDJI
NRP 2512 100 023

Supervisor
Prof. Ir. Budi Santosa M.S. Ph.D

DEPARTMENT OF INDUSTRIAL ENGINEERING
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

LEMBAR PENGESAHAN

IMPLEMENTASI ALGORITMA *HYBRID CROSS ENTROPY – GENETIC ALGORITHM* UNTUK MENYELESAIKAN *SINGLE STAGE CAPACITATED WAREHOUSE LOCATION PROBLEM* (STUDI KASUS : PT. PETROKIMIA GRESIK)

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat Memperoleh Gelar Sarjana Teknik
pada
Program Studi S-1 Jurusan Teknik Industri
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya

Penulis :
FATMAH MUNIF LAHDJI
NRP. 2512 100 023

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Prof. Ir. Budi Santosa M.S. Ph.D
NIP. 196905121994021001

SURABAYA
JANUARI 2016

IMPLEMENTASI ALGORITMA *HYBRID CROSS ENTROPY – GENETIC ALGORITHM* UNTUK MENYELESAIKAN *SINGLE STAGE CAPACITATED WAREHOUSE LOCATION PROBLEM* (STUDI KASUS: PT PETROKIMIA GRESIK)

Nama Mahasiswa : Fatmah Munif Lahdji
NRP : 2512100023
Pembimbing : Prof. Ir. Budi Santosa M.S. Ph.D

ABSTRAK

Single stage capacitated warehouse location problem (SSCWLP) merupakan permasalahan alokasi distribusi dimana produk akan dikirimkan dari *plant* menuju *warehouse* untuk selanjutnya dikirimkan dari *warehouse* menuju pelanggan. Permasalahan SSCWLP ini merupakan permasalahan *NP-hard* dimana penyelesaiannya menggunakan metode eksak membutuhkan waktu yang lama sehingga sering didekati dengan metode metaheuristik. Penelitian ini berfokus pada penentuan alokasi distribusi pupuk PT. Petrokimia Gresik di Pulau Sumatera. Penentuan alokasi distribusi pupuk dilakukan menggunakan algoritma *hybrid cross entropy – genetic algorithm.* Algoritma ini menggabungkan antara mekanisme pembangkitan sampel pada *cross entropy* dengan mekanisme mutasi pada *genetic algorithm* sebagai upaya untuk mempercepat waktu komputasi algoritma *cross entropy* original. Hasil komputasi menggunakan algoritma *hybrid cross entropy genetic algorithm* menunjukkan penghematan biaya distribusi sebesar Rp 737.780.842,00 dengan membuka empat gudang lini 3, antara lain GP Solok, GP Bukittinggi, GP Merangin, dan GP Kotabumi. Berdasarkan hasil eksperimen yang dilakukan, diketahui bahwa secara umum algoritma *hybrid cross entropy – genetic algorithm* mampu menghasilkan alternatif solusi yang lebih mendekati *optimum* dibandingkan algoritma metaheuristik lain seperti algoritma *simulated annealing.* Selain itu, algoritma *hybrid cross entropy – genetic algorithm* juga menghasilkan solusi yang lebih mendekati *optimum* dengan waktu komputasi yang lebih cepat dibangingkan algoritma *cross entropy* original, dan *genetic algorithm* tanpa *crossover.*

Kata Kunci : *Cross Entropy - Genetic Algorithm,* Metaheuristik, *Single Stage Capacitated Warehouse Location Problem*

IMPLEMENTATION OF HYBRID CROSS ENTROPY – GENETIC ALGORITHM FOR SOLVING SINGLE STAGE CAPACITATED WAREHOUSE LOCATION PROBLEM (CASE STUDY: PT PETROKIMIA GRESIK)

Name : Fatmah Munif Lahdji
NRP : 2512100023
Supervisor : Prof. Ir. Budi Santosa M.S. Ph.D

ABSTRACT

Single stage capacitated warehouse location problem (SSCWLP) is an allocation problem in which products will be distributed from the plant to the warehouse for further distributed from the warehouse to the customer. SSCWLP is NP-hard problem, where finding the solution using exact method requires a longer computational time, thus this problem often solved with metaheuristics approach. This study focuses on determining allocation of fertilizer distribution in PT Petrokimia Gresik on Sumatera Island. This allocation problem is computed using a hybrid cross entropy – genetic algorithm method. This algorithm combines the mechanisms of generating samples on cross entropy with mutations mechanism on genetic algorithm in order to speed up the computational time of original cross entropy algorithm. The computational result shows distribution cost savings up to Rp 737.780.842,00 by opening only four *gudang lini 3,* which are GP Solok, GP Bukittinggi, GP Merangin, and GP Kotabumi.Based on the experimental results, it is known that in general, hybrid cross entropy - genetic algorithm is capable of generating an alternative solution which is closer to the optimum compared to other metaheuristic algorithms such as simulated annealing algorithm. In addition, the hybrid cross entropy algorithm - genetic algorithm also generates solution that is both closer to the optimum solution and has faster computational time compared to original cross entropy algorithms and genetic algorithm without crossover.

Key Words : Cross Entropy – Genetic Algorithm, Metaheuristics, Single Stage Capacitated Warehouse Location Problem

KATA PENGANTAR

Puji syukur kehadirat Allah SWT karena atas berkat dan karunia-Nya yang telah diberikan penulis mampu menyelesaikan Tugas Akhir berjudul Implementasi Algoritma *Hybrid Cross Entropy – Genetic Algorithm* dalam Menyelesaikan *Single Stage Capacitated Warehouse Location Problem* (Studi Kasus: PT. Petrokimia Gresik).

Laporan Tugas Akhir ini dibuat sebagai salah satu syarat memperoleh gelar Sarjana pada program studi strata-1 Jurusan Teknik Industri, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember, Surabaya. Penyelesaian laporan ini tidak terlepas dari bantuan banyak pihak, oleh karena itu penulis mengucapkan banyak terima kasih kepada:

1. Munif Mohamad Lahdji dan Jamilah Aboud Bawazeer selaku kedua orang tua penulis yang telah memberikan bantuan, dukungan, doa, dan motivasi yang tidak terhingga kepada penulis
2. Fahd Munif Lahdji, Fachri Munif Lahdji, dan Firza Munif Lahdji selaku kakak kandung penulis yang telah memberikan semangat dan bantuan kepada penulis dengan caranya masing-masing
3. Seluruh keluarga besar penulis yang telah memberi dukungan selama penulis menyelesaikan Tugas Akhir
4. Bapak Nurhadi Siswanto, S.T., M.S.I.E., Ph.D. selaku Ketua Jurusan Teknik Industri, Institut Teknologi Sepuluh Nopember, Surabaya
5. Bapak Prof. Ir. Budi Santosa, M.S. Ph.D. selaku Dosen Pembimbing yang telah memberikan saran, kritik, dan bantuan, serta meluangkan waktu untuk membimbing penulis dalam menyelesaikan Tugas Akhir
6. Bapak Yudha Andrian Saputra, S.T., MBA. selaku Koordinator Tugas Akhir atas kelancaran selama proses penyusunan tugas akhir
7. Aditya Cahya Wardhana, S.T. selaku pembibing penulis di Departemen Distribusi Wilayah II, PT. Petrokimia Gresik dan segenap karyawan PT. Petrokimia Gresik lainnya yang telah banyak membantu dan memberikan

kemudahan bagi penulis dalam pengumpulan data dan penyelesaian Tugas Akhir

8. Segenap dosen dan karyawan Jurusan Teknik Industri ITS yang telah banyak memberikan pelajaran dan pengalaman bagi penulis selama menempuh studi di Jurusan Teknik Industri ITS
9. Keluarga besar Kavaleri 2012 atas pengalaman dan kebersamaan dalam suka duka selama menempuh studi di Jurusan Teknik Industri ITS
10. Dini, Tia, Riris, Niela, Ocha, Agung, Yuni, Sari, Nurinda, dan pejuang-pejuang #113 lainnya yang saling memberi semangat, dukungan, motivasi dalam menyelesaikan penulisan Tugas Akhir ini
11. Seluruh teman dekat penulis yang tidak dapat disebutkan satu-persatu atas segala dukungannya

Serta berbagai pihak yang tidak dapat penulis sebutkan satu per satu. Semoga Allah SWT membalas semua kebaikan yang telah dilakukan.

Penulis menyadari bahwa masih banyak kekurangan dalam Tugas Akhir ini mengingat kurangnya pengalaman dan pengetahuan penulis. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan sebagai motivasi dalam rangka pengembangan diri menjadi lebih baik.

Surabaya, Januari 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

BAB 2 TINJAUAN PUSTAKA

BAB 3 METODOLOGI PENELITIAN

BAB 4 PENGEMBANGAN MODEL DAN ALGORITMA

BAB 5 EKSPERIMEN DAN ANALISIS

DAFTAR TABEL

BAB 2 TINJAUAN PUSTAKA

BAB 4 PENGEMBANGAN MODEL DAN ALGORITMA

BAB 5 EKSPERIMEN DAN ANALISIS

BAB 1
PENDAHULUAN

Bab pendahuluan ini berisi hal-hal yang mendasari dilakukannya penelitian serta identifikasi masalah penelitian. Bahasan yang terdapat pada bab pendahuluan ini meliputi latar belakang masalah, perumusan masalah, ruang lingkup penelitian, tujuan penelitian, dan manfaat penelitian.

1.1 Latar Belakang

Tantangan dalam dunia industri mengalami pergerakan dari masa ke masa, hingga akhirnya sejak tahun 1990-an pelaku industri mulai sadar terhadap pentingnya aspek kecepatan respon, inovasi, dan fleksibilitas dalam menanggapi persaingan. Pemenuhan ketiga aspek tersebut membutuhkan peran serta semua pihak terkait mulai dari *supplier,* pabrik, perusahaan transportasi, hingga jaringan distribusi yang akan menyalurkan produk ke pelanggan atau yang lebih dikenal dengan konsep *supply chain management* (Pujawan, 2005). Menurut Chopra dan Meindl (2001) agar suatu perusahaan dapat mencapai keseimbangan antara *responsiveness* dan efisiensi agar proses pemenuhan *demand* pelanggan lebih efektif dan efisien, pelaku industri harus mempertimbangkan keempat *driver* dalam *supply chain,* yaitu fasilitas, persediaan, transportasi, dan informasi. Salah satu *driver* yang memiliki peran penting dalam *supply chain* ini adalah fasilitas, dimana keputusan mengenai fasilitas seperti lokasi, kapasitas, dan fleksibilitasnya memiliki pengaruh yang signifikan terhadap performansi *supply chain* perusahaan secara keseluruhan. Pertimbangan yang tepat pada lokasi dan jumlah fasilitas akan memengaruhi tingkat respon terhadap pelanggan serta efisiensi biaya distribusi perusahaan.

Salah satu permasalahan dalam penentuan fasilitas ini adalah bagaimana menentukan lokasi fasilitas beserta alokasinya dengan tujuan untuk mendapatkan biaya distribusi yang minimum atau lebih dikenal dengan istilah *facility location problem* (FLP). Menurut Peng, Rui-hua, dan Jin, 2008, permasalahan FLP klasik ini merupakan *NP-hard problem* yang memiliki tingkat kompleksitas tinggi sehingga banyak dilakukan penelitian untuk mengembangkan model dan

algoritma penyelesainnya, antara lain *gravity methods, mixed integer linear programming,* dan metode metaheuristik seperti *simulated annealing* dan *genetic algorithm.*

Pengembangan terhadap permasalahan *facility location* juga banyak dilakukan, salah satunya adalah *capacitated k-facility location problem* (CKLP). Dalam permasalahan CKLP, peneliti diberikan sejumlah *D* pelanggan dan sejumlah *F* fasilitas potensial dimana setiap fasilitas *i* memiliki kapasitas sebesar s_i, dan setiap pelanggan *j* memiliki permintaan sebesar *dj* yang harus dilayani. Fungsi objektif dari permasalahan ini adalah untuk dapat memenuhi permintaan pelanggan dengan menggunakan paling banyak sejumlah *k* fasilitas tanpa melanggar batasan kapasitas sehingga dihasilkan total biaya seminimal mungkin (Aardal, Berg, Gijswijt, dan Li, 2015). Menurut Pál et al. 2001 dalam Aardal et al. 2015, penyelesaian CKLP sebagian besar diselesaikan dengan *local search technique* yaitu dengan *approximation algorithm* karena secara umum penyelesaian menggunakan relaksasi *linear programming* menunjukkan *unbounded integrality gap.*

Varian lain yang dikembangkan dari FLP salah satunya adalah *warehouse location problem* (WLP). WLP merupakan pengembangan dari FLP yang digunakan dalam perencanaan lokasi gudang distribusi untuk menghindari *potential losses* yang diakibatkan oleh pemilihan lokasi yang tidak optimal. Penelitian mengenai WLP telah banyak dikembangkan, salah satunya adalah penyelesaian WLP dengan metode eksak yang dilakukan oleh Khumalawa pada tahun 1972 menggunakan metode *branch and bound.* Selain itu, juga dilakukan beberapa pengembangan terhadap model permasalahan WLP, salah satunya adalah *single stage capacitated warehouse location problem* (SSCWLP). Pada permasalahan SSCWLP, produk akan dikirimkan dari *plant* menuju *warehouse* untuk selanjutnya dikirimkan dari *warehouse* menuju pelanggan (Sharma dan Berry, 2007). Permasalahan SSCWLP ini memiliki tingkat kompleksitas yang serupa dengan permasalahan FLP yaitu *NP-hard* dimana permasalahan ini sulit untuk diselesaikan dengan metode eksak sehingga sering didekati dengan metode metaheuristik seperti *particle swarm optimization, genetic algorithm, tabu search,* dan *simulated annealing.* Penggunaan metode metaheuristik mampu

menunjukkan solusi yang cukup baik atau dekat dengan solusi optimal dalam waktu komputasi yang relatif singkat dibandingkan penyelesaian menggunakan metode eksak.

Permasalahan yang akan diteliti pada penelitian ini merupakan pengembangan dari penelitian yang telah dilakukan oleh Kresna (2014). Pada penelitian sebelumnya, dilakukan penyelesaian permasalahan *single stage capacitated warehouse location problem* dengan studi kasus di PT. Petrokimia Gresik dengan menggunakan algoritma *simulated annealing.* Dalam penelitian sebelumnya, dilakukan pemilihan lokasi gudang penyangga PT. Petrokimia Gresik dan alokasi *demand*-nya untuk meminimasi biaya total distribusi dan sewa *warehouse* dibandingkan terhadap kondisi eksisting distribusi pupuk di daerah distribusi wilayah II. Pada kondisi eksisting diketahui bahwa PT. Petrokimia Gresik menyewa 30 unit gudang penyangga atau gudang lini 3 di Pulau Sumatera dan distributor dapat mengambil pupuk secara langsung di gudang lini II maupun gudang lini III. Pertimbangan utama untuk penetapan gudang lini 3 tersebut adalah kedekatan dengan lokasi *demand* atau distributor sebagai konsumen akhir dari PT. Petrokimia Gresik. Akan tetapi belum diketahui apakah lokasi dan jumlah gudang penyangga tersebut memberikan biaya yang minimum dari segi biaya distribusi. Berdasarkan kondisi tersebut maka telah dilakukan penelitian dengan cara merubah mekanisme distribusi pupuk PT. Petrokimia Gresik di Pulau Sumatera dengan cara mengurangi jumlah gudang lini III yang disewa namun agar tidak merugikan distributor dengan mengurangi kedekatan dengan lokasi gudang maka pengiriman dari gudang lini II maupun gudang lini III ke lokasi distributor dibebankan kepada PT. Petrokimia Gresik.

Berdasarkan penelitian tersebut, didapatkan bahwa solusi yang dihasilkan dari implementasi algoritma *simulated annealing* mampu memberikan penghematan sebesar Rp 689.236.874,00 bagi PT. Petrokimia Gresik. Namun, penghematan yang ditunjukkan masih relatif jauh dibandingkan solusi optimum yang didapatkan dari perhitungan dengan metode eksak, yaitu dengan perbedaan sebesar Rp 118.860.000,00.. Data tersebut menunjukkan adanya *potential losses* yang cukup besar dalam penyelesaian SSCWLP untuk *scope* permasalahan menengah. Oleh karena itu, peneliti mencoba untuk menyelesaikan permasalahan

single stage capacitated warehouse location problem dengan menggunakan algoritma *hybrid cross entropy* dan *genetic algorithm* dengan harapan mampu meminimasi *potential losses* yang dihasilkan.

Menurut Rubinstein (1997) dalam Santosa dan Willy (2011), algoritma *cross entropy* pada awalnya merupakan penerapan algoritma stokastik kompleks yang digunakan untuk mengestimasi probabilitas *rare event* atau kejadian langka. Namun, pada perkembangan selanjutnya algoritma ini mengalami modifikasi sederhana sehingga dapat digunakan untuk menyelesaikan permasalahan optimasi kombinatorial dengan cara menerjemahkan masalah optimasi deterministik menjadi stokastik. *Cross entropy* telah berhasil diimplementasikan untuk menyelesaikan berbagai permasalahan seperti *travelling salesman problem* (TSP), *quadratic assignment problem, buffer allocation problem* (BAP), *vehicle routing problem* (VRP), dan lain-lain (Santosa dan Hardiansyah, 2010). Akan tetapi, salah satu permasalahan yang sering dihadapi dalam aplikasi CE adalah waktu komputasi yang cukup lama. Untuk mengurangi waktu komputasi tersebut, algoritma CE dapat di-*hybrid* dengan algortima metaheuristik yang lain, seperti *genetic algorithm.*

Genetic algorithm merupakan teknik metaheuristik yang dikembangkan berdasarkan mekanisme seleksi alam dan pewarisan genetik. Dalam penerapan algoritma ini dilakukan beberapa mekanisme stokastik, antara lain pembangkitan populasi, elitisme, *crossover,* dan mutasi. Akan tetapi, *hybrid* CE dan GA ini dilakukan dengan tujuan untuk mempercepat waktu komputasi, sehingga dalam penelitian ini mekanisme *crossover* akan dihilangkan karena akan membutuhkan koreksi yang mengakibatkan waktu komputasi bertambah lama dan memungkinkan individu yang dibangkitkan semakin menjauh dari nilai optimal. *Hybrid cross entropy* dan *genetic algorithm* telah sukses diterapkan untuk kasus penjadwalan, salah satunya adalah dalam *multi objective job shop scheduling problem* (Santosa dan Nurkhalida, 2012). Selain itu, algortima *hybrid* CE dan GA ini juga telah sukses diterapkan dalam penyelesaian *multi-product inventory ship routing problem with heterogeneous fleet* (Santosa dan Damayanti, 2014). Oleh karena itu, pada penelitian ini akan digunakan algoritma *hybrid* CEGA untuk memperoleh hasil yang lebih baik dalam penyelesaian SSCWLP.

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang yang telah dijelaskan sebelumnya, permasalahan pada penelitian ini adalah bagaimana mengembangkan algoritma *hybrid cross entropy* dan *genetic algorithm* dalam menyelesaikan permasalahan *single stage capacitated warehouse location problem* dengan studi kasus di PT. Petrokimia Gresik.

1.3 Ruang Lingkup Penelitian

Pada bagian ini akan dijelaskan mengenai batasan dan asumsi yang digunakan dalam penelitian ini.

1.3.1 Batasan

Berikut merupakan batasan-batasan yang digunakan dalam penelitian ini, antara lain:

1. Permasalahan yang diteliti adalah *single stage capacitated warehouse location problem* (SSCWLP)
2. Penelitian dilakukan terhadap gudang penyangga PT. Petrokimia Gresik distribusi wilayah II di Pulau Sumatera
3. Gudang yang diamati pada penelitian ini adalah gudang lini dua dan gudang lini tiga

1.3.2 Asumsi

Berikut merupakan asumsi-asumsi yang digunakan dalam penelitian ini, antara lain:

1. Seluruh *input* data bersifat deterministik
2. Setiap lokasi *demand* hanya dapat dipasok dari satu gudang
3. Lokasi *demand* berada di pusat kabupaten/kota
4. Kapasitas moda transportasi menuju lokasi *demand* adalah 15 ton

1.4 Tujuan Penelitian

Tujuan penelitian yang ingin dicapai dalam penelitian tugas akhir ini adalah sebagai berikut:

1. Mengimplementasikan algoritma *hybrid cross entropy - genetic algorithm* dalam menyelesaikan *single stage capacitated warehouse location problem*
2. Memberikan alternatif gudang penyangga beserta alokasi distribusi pupuk bagi PT Petrokimia Gresik dengan biaya yang minimum
3. Menguji dan membandingkan algoritma *hybrid cross entropy - genetic algorithm* dengan algoritma *simulated annealing, cross entropy,* dan *modified genetic algorithm*

1.5 Manfaat Penelitian

Manfaat yang ingin diperoleh dalam penelitian ini adalah sebagai berikut:

1. Mengisi *gap* penelitian dalam permasalahan *single stage capacitated warehouse location problem*
2. Memberikan alternatif gudang penyangga bagi PT. Petrokimia Gresik
3. Memberikan kontribusi dalam bidang keilmuan optimasi, yaitu dengan implementasi algoritma *hybrid cross entropy - genetic algorithm* dalam menyelesaikan *single stage capacitated warehouse location problem*

1.6 Sistematika Penulisan

Laporan tugas akhir ini terdiri dari enam bab dengan sistematika penulisan sebagai berikut:

BAB I PENDAHULUAN

Pada bab pendahuluan akan dijelaskan mengenai hal-hal yang mendasari dilakukannya penelitian dan identifikasi masalah penelitian. Bahasan yang terdapat pada bab pendahuluan ini meliputi latar belakang masalah, perumusan masalah, ruang lingkup penelitian, tujuan penelitian, manfaat penelitian, serta sistematika penulisan

BAB II TINJAUAN PUSTAKA

Tinjauan pustaka menguraikan teori, temuan, dan bahan penelitian lain yang diperoleh dari acuan yang akan dijadikan landasan untuk melakukan kegiatan penelitian yang akan dijadikan tugas akhir. Pada bab tinjauan pustaka akan dijelaskan mengenai *facility location problem, warehouse*

location problem, single stage capacitated warehouse location problem, algoritma *cross entropy,* dan *genetic algorithm.*

BAB III METODOLOGI PENELITIAN

Bab ini menguraikan metodologi penelitian yang dilakukan untuk menyelesaikan permasalahan *single stage capacitated warehouse location problem* dengan studi kasus di PT. Petrokimia Gresik dengan menggunakan algoritma *hybrid cross entropy* dan *genetic algorithm.*

BAB IV PENGEMBANGAN MODEL DAN ALGORITMA

Pada bab ini akan dijelaskan bagaimana mengembangkan algotirma *hybrid cross entropy* dan *genetic algorithm* untuk penyelesaian permasalahan *single stage capacitated warehouse location problem.*

BAB V EKSPERIMEN DAN ANALISIS

Bab ini meliputi pengujian algoritma sesuai dengan pengembangan algoritma yang telah dibuat dan analisis mengenai hasil eksperimen yang telah dilakukan.

BAB VI KESIMPULAN DAN SARAN

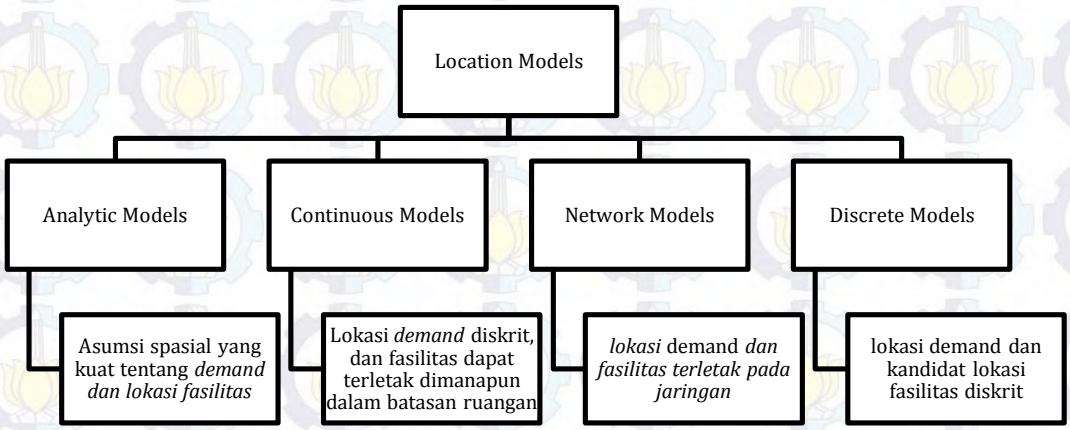
Bab ini berisi tentang kesimpulan hasil penelitian dan saran-saran yang berkaitan dengan penelitian selanjutnya.

BAB 2
TINJAUAN PUSTAKA

Bab tinjauan pustaka menguraikan teori, temuan, dan bahan penelitian lain yang diperoleh dari acuan yang akan dijadikan landasan untuk melakukan kegiatan penelitian yang akan dijadikan tugas akhir. Pada bab tinjauan pustaka akan dijelaskan mengenai *facility location problem, warehouse location problem, single stage capacitated warehouse location problem,* algoritma *cross entropy,* dan *genetic algorithm.*

2.1 *Facility Location Problem*

Facility location problem merupakan permasalahan dalam menentukan lokasi optimal untuk suatu set fasilitas dengan adanya permintaan yang stokastik. Permasalahan utama dalam model ini adalah untuk menentukan lokasi terbak untuk suatu set fasilitas dengan menentukan kapasitas pelayanan pada setiap fasilitas. (Berman dan Krass, 2002). Menentukan lokasi fasilitas dalam suatu jaringan *supply chain* merupakan permasalahan keputusan penting yang memberikan pengaruh terhadap bentuk dan struktur dalam sistem *supply chain* secara keseluruhan. Perancangan ini menetapkan alternatif-alternatif beserta biaya terkait dan level investasi yang digunakan dalam mengoperasikan sistem. Keputusan lokasi melibatkan penentuan jumlah, lokasi, dan ukuran dari fasilitas yang akan digunakan. Fasilitas-fasilitas tersebut antara lain adalah pabrik, pelabuhan, *vendors,* gudang, *retail outlet, service centers,* dan lain-lain yang merupakan titik dalam jaringan *supply chain* dimana produk secara sementara terhenti dalam perjalanannya menuju konsumen akhir (Ballou, 2003). Oleh karena itu, keputusan lokasi yang buruk akan mengakibatkan peningkatan biaya dan menurunkan *competitiveness.*



Gambar 2. 1 Alternatif Taksonomi Model Lokasi (Sumber: Daskin, 2013)

Gambar 2.1 menunjukkan taksonomi dari model lokasi yang terdiri dari empat jenis, yaitu *analytic location models, continuous location models, network location models,* dan *discrete location models.* Pada *analytic location models* permintaan diasumsikan terdistribusi dalam beberapa cara dalam ruang, sedangkan kandidat fasilitas dapat terletak dimanapun dalam wilayah pelayanan.

Pada *contionuous location models* permintaan diasumsikan muncul pada lokasi diskrit dan level permintaan adalah *a priori,* sedangkan kandidat fasilitas dapat terletak dimanapun dalam wilayah pelayanan. Salah satu model yang banyak dikenal dalam *continuous location model* adalah weber location model dimana dalam model ini dintentukan lokasi *center of gravity* pada titik permintaan.

Network location models memperlakukan fasilitas dan lokasi *demand* terletak pada jaringan yang terdiri dari *nodes* dan *link* atau *arc.* Dalam model ini, umumnya *demand* terletak pada *nodes* dalam jaringan, sedangkan lokasi fasilitas dapat diletakkan pada *nodes* atau *arc* yang terdapat dalam jaringan.

Pada *discrete location models* tidak ada asumsi khusus yang ditambahkan mengenai lokasi *demand* dan kandidat fasilitas. Pada model ini, secara sederhana diketahui lokasi atau koordinat titik *demand* dan alternatif fasilitas. Selain itu, jarak antara fasilitas dan *demand* tidak perlu diselesaikan dengan formulasi khusus. *Discrete location models* sering diformulasikan dalam bentuk *integer*

programming dan diselesaikan menggunakan metode eksak maupun metode heuristik.

2.1.1 *Uncapacitated Facility Location Problem*

Uncapacitated facility location problem (UFLP) merupakan permasalahan dalam menentukan jumlah dan lokasi fasilitas untuk meminimasi total biaya baik berupa *fixed cost* maupun *variable cost.* UFLP lebih menekankan terhadap distribusi satu jenis produk dalam satu periode waktu dimana *demand* yang ada diasumsikan deterministik. Dalam permasalahan ini, lokasi alternatif fasilitas dan lokasi *demand* merupakan titik diskrit yang sudah ditentukan, sedangkan kapasitas dari fasilitas yang akan dibangun diasumsikan tidak terbatas (Eiselt dan Marianov, 2011). Berikut merupakan model yang digunakan dalam permasalahan UFLP, yaitu:

- **Parameter:**

  i = alternatif lokasi fasilitas (1, 2, …, m)

  j = konsumen (1, 2, …, n)

  f_i = *fixed cost* dalam penempatan fasilitas i

  c_{ij} = total biaya produksi dan pengiriman dari fasilitas i ke konsumen j

  d_j = *demand* konsumen j

- **Variabel Keputusan:**

  x_{ij} *:* jumlah *demand* konsumen j yang dipenuhi dari fasilitas i

  y_i *:* variabel biner yang menunjukkan keputusan mendirikan fasilitas, bernilai 1 apabila fasilitas i didirikan dan bernilai 0 apabila fasilitas i tidak didirikan

- **Fungsi Tujuan:**

  Fungsi tujuan dari UFLP adalah untuk meminimasi total biaya produksi dan pengiriman serta pendirian fasilitas. Biaya produksi dan pengiriman dianggap sebagai fungsi linear yang sebanding dengan kuantitas produk yang dikirim dari setiap fasilitas.

  $$Min\ z = \sum_i \sum_j C_{ij} x_{ij} + \sum_i f_i y_i \qquad (2.1)$$

- **Batasan:**

  $$\sum_i x_{ij} = d_j \quad , \forall j \qquad (2.2)$$

$$x_{ij} \leq y_i \quad ,\forall i,j \tag{2.3}$$

$$x_{ij} \geq 0 \quad ,\forall i,j \tag{2.4}$$

$$y_i \in \{0,1\} \quad ,\forall \boldsymbol{i} \tag{2.5}$$

Persamaan (2.2) memastikan bahwa seluruh *demand* untuk setiap konsumen dapat dipenuhi. Persamaan (2.3) menunjukkan bahwa *demand* j hanya dapat dipenuhi melalui fasilitas i apabila fasilitas i didirikan. Sedangkan persamaan (2.4) menunjukkan batasan *non-negativity*.

2.1.2 *Capacitated Facility Location Problem*

Sama seperti model UFLP, model *capacitated facility location problem* (CFLP) juga merupakan permasalahan dalam menentukan jumlah dan lokasi fasilitas untuk meminimasi total biaya baik berupa *fixed cost* maupun *variable cost.* Namun, perbedaan antara kedua model ini adalah pada model CFLP fasilitas yang akan didirikan memiliki kapasitas tertentu yang sudah ditentukan, berbeda dengan pada model UFLP dimana kapasitas diasumsikan tidak terbatas. Model matematis CFLP serupa dengan model matematis UFLP namun terdapat tambahan satu parameter dan satu batasan terkait dengan kapasitas fasilitas. Berikut merupakan tambahan parameter dalam model permasalahan CFLP, yaitu:

- **Parameter:**

  k_i = kapasitas fasilitas *i*

- **Batasan:**

  $$\sum_j x_{ij} \leq k_i \quad ,\forall i \tag{2.6}$$

  Persamaan (2.6) menunjukkan bahwa jumlah produk yang dikirimkan dari fasilitas *i* ke konsumen tidak melebihi kapasitas dari fasilitas *i.*

2.2 *Warehouse Location Problem*

Salah satu fasilitas yang ditentukan dalam *facility location problem* adalah *warehouse* atau gudang. *Warehouse* merupakan bangunan komersial yang digunakan sebagai tempat penyimpanan dan distribusi produk. Gudang merupakan dalah satu fungsi penting dalam aliran barang di sepanjang *supply chain* yang dapat menciptakan keunggulan dari sisi kecepatan waktu, kualitas, dan

efisiensi. Menurut Ballou (2003), secara umum terdapat empat alasan mendasar dalam penggunaan *warehouse,* antara lain:

1. Reduksi Biaya Transportasi-Produksi

   Warehousing dan inventori terkait secara umum memang meningkatkan pengeluaran, akan tetapi mungkin terdapat *trade-off* dengan biaya yang lebih rendah yang didapatkan dari peningkatan efisiensi dalam transportasi dan produksi
2. Koordinasi antara *Supply* dan *Demand*

   Pertimbangan harga komoditas juga dapat menghasilkan kebutuhan untuk penggunaan gudang. Material dan produk yang memiliki perubahan harga yang besar dari waktu ke waktu mendorong perusahaan untuk membeli komoditas tersebut sebelum kebutuhan mereka terhadap barang tersebut muncul untuk mendapatkan barang dengan harga yang lebih rendah. Dalam hal ini terdapat *trade-off* antara biaya penyimpanan produk dan penghematan dari harga komoditas
3. Kebutuhan Produksi

   Terkadang *warehouse* juga digunakan sebagai tempat dalam melakukan aktivitas *value added* seperti *packaging, labeling,* maupun proses penyimpanan yang merupakan salah satu aktivitas produksi seperti halnya penyimpanan keju dan *wine*
4. Pertimbangan Aspek Pemasaran

   Pemasaran sering terkait dengan bagaimana ketersediaan produk di pasar. Penggunaan *warehouse* dapat digunakan untuk mendekatkan produk ke konsumen sehingga *delivery time* dapat diturunkan dan avalibilitas produk di pasar meningkat sehingga mampu memicu peningkatan penjualan

2.3 *Capacitated k-Facility Location Problem*

Pada *capacitated k-facility location problem* (CKFL) peneliti memiliki data input sejumlah *D* pelanggan dan sejumlah *F* fasilitas potensial dimana setiap fasilitas *i* memiliki kapasitas sebesar s_i, dan setiap pelanggan *j* memiliki permintaan sebesar *dj* yang harus dilayani. Fungsi objektif dari permasalahan ini

adalah untuk dapat memenuhi permintaan pelanggan dengan menggunakan paling banyak sejumlah *k* fasilitas tanpa melanggar batasan kapasitas sehingga dihasilkan total biaya seminimal mungkin. Permasalahan ini dapat diformulasikan dalam bentuk *mixed integer program* sebagai berikut (Aardal et. al, 2015).

- **Parameter:**

  i = fasilitas

  j = konsumen

  c_{ij} = biaya pengiriman dari i ke j

  d_j = permintaan tiap konsumen j

  f_i = *fixed cost* penempatan fasilitas ke-i

  s_i = kapasitas fasilitas

  k = jumlah fasilitas yang akan didirikan

- **Variabel Keputusan:**

  x_{ij} : jumlah *demand* konsumen j yang dipenuhi dari fasilitas i

  y_i : variabel biner yang menunjukkan keputusan mendirikan fasilitas, bernilai 1 apabila fasilitas i didirikan dan bernilai 0 apabila fasilitas i tidak didirikan

- **Fungsi Tujuan:**

  Fungsi tujuan dari CKLP adalah untuk meminimasi total biaya produksi dan pengiriman serta pendirian fasilitas. Biaya produksi dan pengiriman dianggap sebagai fungsi linear yang sebanding dengan kuantitas produk yang dikirim dari setiap fasilitas.

  $$Min\ z = \sum_i \sum_j C_{ij} x_{ij} + \sum_i f_i y_i \quad (2.7)$$

- **Batasan:**

  $$\sum_i x_{ij} = d_j \quad ,\forall j \quad (2.8)$$

  $$\sum_j x_{ij} \leq s_i y_i \quad ,\forall i \quad (2.9)$$

  $$\sum_i y_i \leq k \quad (2.10)$$

  $$x_{ij} \geq 0 \quad ,\forall i,j \quad (2.11)$$

  $$y_i \in \{0,1\} \quad ,\forall i \quad (2.12)$$

  Persamaan (2.8) memastikan bahwa seluruh permintaan pelanggan terpenuhi. Persamaan (2.9) memastikan bahwa jumlah produk yang

dikirimkan dari fasilitas *i* tidak melebih kapasitas fasilitas *i* dan permintaan konsumen *j* hanya dapat dipenuhi dari fasilitas *i* jika fasilitas *i* didirikan. Sedangkan persamaan (2.10) menunjukkan bahwa jumlah maksimal fasilitas yang dapat didirikan sejumlah *k* fasilitas.

2.4 *Single Stage Capacitated Warehouse Location Problem*

Single stage capacitated warehouse location problem (SSCWLP) merupakan spesialisasi dari model *warehouse location problem* dimana dalam model ini produk akan dikirimkan dari *plant* menuju *warehouse* untuk selanjutnya dikirimkan dari *warehouse* menuju pelanggan. Permasalahan dalam model ini adalah untuk memilih sejumlah titik dimana *warehouse* akan diletakkan sehingga jumlah biaya gudang dan biaya transportasi dapat diminimumkan Berikut merupakan model matematis dari permasalahan *single stage warehouse location problem* (Sharma dan Berry, 2007).

- **Parameter:**

  i = *plant*

  j = *warehouse*

  k = pasar

  D_k = komoditas permintaan pasar k

  $d_k = D_k/\Sigma D_k$ permintaan pasar k sebagai fraksi dari total permintaan pasar

  S_i = *supply* yang tersedia pada *plant i*

  $s_i = S_i/\Sigma D_k$ *supply* yang tersedia pada *plant i* sebagai fraksi dari total permintaan pasar

  f_i = *fixed cost* dalam penempatan fasilitas i

  C_{ijk} = biaya dalam pengiriman sejumlah ΣDk produk dari lokasi *plant i* ke *warehouse j* ke lokasi pasar k

  CAP_j = kapasitas *warehouse j*

  $cap_j = CAP_j/\Sigma D_k$ kapasitas *warehouse j* sebagai fraksi dari total permntaan pasar

- **Variabel Keputusan:**

  X_{ijk} = kuantitas komoditas yang dikirimkan dari *plant i* menuju *warehouse j* menuju pasar k

x_{ijk} = $X_{ijk}/\Sigma D_k$ kuantitas yang dikirimkan sebagai fraksi dari total permintaan pasar

y_j = variabel biner yang menunjukkan keputusan mendirikan fasilitas, bernilai 1 apabila *warehouse j* didirikan dan bernilai 0 apabila *warehouse j* tidak didirikan

- **Fungsi Tujuan:**

Fungsi tujuan dari *single stage capacitated warehouse location problem* adalah untuk meminimasi total biaya distribusi serta pendirian fasilitas sebagaimana ditunjukkan pada persamaan (2.11) berikut.

$$Min\ z = \sum_i \sum_j \sum_k C_{ijk} x_{ijk} + \sum_i f_i y_i \quad (2.11)$$

- **Batasan:**

$$\sum_i \sum_j \sum_k x_{ijk} = 1 \quad (2.12)$$

$$\sum_j \sum_k x_{ijk} \leq s_i,\ \forall i \quad (2.13)$$

$$\sum_i \sum_j x_{ijk} \geq d_k,\ \forall k \quad (2.14)$$

$$\sum_i \sum_k x_{ijk} \leq \text{cap}_j,\ \forall j \quad (2.15)$$

$$x_{ijk} \geq 0,\ \forall i, j, k \quad (2.16)$$

Persamaan (2.12) memastikan bahwa seluruh aliran produk melalui jaringan berjumlah sama dengan total permintaan seluruh pasar. Persamaan (2.13) memastikan bahwa aliran produk yang keluar dari lokasi *supply* tidak melebihi ketersediaan *supply* yang ada. Persamaan (2.14) menunjukkan bahwa aliran produk yang memasuki pasar mampu memenuhi permintaan pasar. Sedangkan persamaan (2.16) merupakan batasan *non-negativity* (Sharma dan Berry, 2007).

Selanjutnya merupakan batasan-batasan yang mengaitkan antara variabel *real* dengan variabel biner, antara lain:

$$\sum_i \sum_k x_{ijk} \leq \text{cap}_j y_j,\ \forall j \quad (2.17)$$

$$\sum_i \sum_k x_{ijk} \leq y_j,\ \forall j \quad (2.18)$$

$$\sum_i x_{ijk} \leq d_k y_j,\ \forall j, k \quad (2.19)$$

$$\sum_k x_{ijk} \leq s_i y_j,\ \forall i, j \quad (2.20)$$

$$\sum_i \sum_k x_{ijk} + M(1 - y_j) \geq 0,\ \forall j$$

$$\sum_i \sum_k x_{ijk} + My_j \geq 0, \ \forall j \quad (2.21)$$

$$\sum_i \sum_k x_{ijk} - My_j \leq 0, \ \forall j$$

$$\sum_i x_{ijk} - M(1 - y_j) \leq d_k, \ \forall j, k$$

$$\sum_i x_{ijk} + My_j \geq 0, \ \forall j, k \quad (2.22)$$

$$\sum_i x_{ijk} - My_j \leq 0, \ \forall j, k$$

$$\sum_k x_{ijk} - M(1 - y_j) \leq d_k, \ \forall i, j$$

$$\sum_k x_{ijk} + My_j \geq 0, \ \forall i, j \quad (2.23)$$

$$\sum_k x_{ijk} - My_j \leq 0, \ \forall i, j$$

$$y_j = (0, 1), \ \forall j \quad (2.24)$$

$$y_j \geq 0, \ \forall j \quad (2.25)$$

2.5 Algoritma *Cross Entropy*

Menurut Rubinstein (1997) dalam Santosa dan Willy (2011), algoritma *cross entropy* pada awalnya merupakan penerapan algoritma stokastik kompleks yang digunakan untuk mengestimasi probabilitas *rare event* atau kejadian langka. Namun, pada perkembangan selanjutnya algoritma ini mengalami modifikasi sederhana sehingga dapat digunakan untuk menyelesaikan permasalahan optimasi kombinatorial dengan cara menerjemahkan masalah optimasi deterministik menjadi stokastik.

Menurut Santosa dan Willy (2011), dalam algoritma *cross entropy* dilakukan pembangkitan sejumlah sampel seperti halnya Monte Carlo. Selanjutnya akan dilakukan pembangkitan sampel random yang dibangkitkan berdasarkan parameter yang didapat dari sampel sebelumnya. Ilustrasi dari metode *cross entropy* dapat dijelaskan sebagai berikut: misal terdapat suatu masalah minimasi fungsi *f(x)* pada setiap *x* yang berasal dari *X* dimana nilai minimum fungsi tersebut adalah γ*.

$$\gamma * = \min_{x \in X} f(x) \quad (2.26)$$

Karena nilai *x* dan γ* belum diketahui, maka perlu dibangkitkan beberapa bilangan random *x* melalui suatu *probability density function* (pdf) tertentu. Misalnya, bangkitkan nilai *x* berdistribusi normal sejumlah N sampel, dimana dalam pembangkitan distribusi normal dibutuhkan parameter μ dan σ untuk

membangkitkan *X*. tujuan algoritma ini adalah untuk menghasilkan urutan solusi $\{(\gamma_t, v_t)\}$ yang dengan cepat dapat memusat ke solusi optimal (γ^*, v^*) dimana γ^* merupakan fungsi tujuan yang dicari, v^* merupakan parameter pdf, dan t adalah iterasi. untuk langkah awal harus ditetapkan nilai v_0 dan nilai parameter ρ yang tidak terlalu kecil (misal $\rho = 0.1$). Parameter ini digunakan untuk menyatakan sampel elite atau persentase sampel keseluruhan yang akan dipilih untuk meng-*update* parameter v yang digunakan. Selain itu juga diperlukan konstanta α yang digunakan untuk membobotkan parameter pada iterasi sekarang dan iterasi sebelumnya.

Tahapan secara matematis dari metode *cross entropy* dapat dinyatakan sebagai berikut:

1. Tentukan nilai parameter awal berupa N (jumlah sampel), v_0 (μ_0, dan σ_0), ρ, dan α
2. Bangkitkan sampel sebanyak N dengan memanfaatkan parameter v_0
3. Evaluasi *fitness function* untuk setiap sampel lalu urutkan dari yang terkecil hingga terbesar
4. Pemilihan sampel elite sebanyak ρ persentil dari urutan sampel yang didapatkan
5. *Update* parameter v dengan persamaan (2.27) dimana $\tilde{w}_t$ merupakan parameter vektor yang didapatkan dari solusi sampel elite dan α merupakan parameter *smoothing* dengan $0.7 < \alpha < 1$

$$\hat{v}_t = \propto \tilde{w}_t + (1-\propto)\hat{v}_{t-1} \qquad (2.27)$$

2.6 *Genetic Algorithm*

Genetic algorithm merupakan algoritma metaheuristik yang dikembangkan oleh John Holland berdasarkan teori mekanisme seleksi alam dan pewarisan genetik Charles Darwin. Algoritma ini pada awalnya dikembangkan untuk menyelesaikan permasalahan kontinu, namun telah banyak dikembangkan untuk menyelesaikan permasalahan diskrit. Dalam algoritma ini, populasi solusi dipertahankan oleh operator mutasi dan *crossover* dimana operator *crossover* mensimulasikan reproduksi. Kualitas setiap solusi ditunjukkan oleh nilai *fitness*-nya, dimana nila ini digunakan untuk memilih sebuah solusi dalam populasi untuk

reproduksi dan ketika solusi dikeluarkan dari populasi. Kualitas rata-rata populasi secara bertahap menigkat sebagaimana solusi yang baru dan lebih baik dibangkitkan dan solusi yang buruk dihilangkan (Razali, 2015).

Secara umum, langkah-langkah metode *genetic algorithm* adalah sebagai berikut:

1. Bangkitkan populasi kandidat awal solusi sebanyak N
2. Lakukan evaluasi *fitness function* untuk masing-masing individu atau solusi menggunakan persamaan (2.28) dimana *F(x)* menunjukkan *fitness function* dan *f(x)* menunjukkan fungsi tujuan (Santosa dan Willy, 2011)

   $$F(x) = \frac{1}{1+f(x)} \quad (2.28)$$

3. Elitisme, pemilihan individu terbaik untuk disalin sejumlah tertentu sebagai usaha untuk menjaga individu terbaik tetap muncul dalam populasi pada iterasi berikutnya
4. Seleksi, pemilihan anggota populasi yang akan dijadikan induk dalam proses *crossover*. Proses seleksi ini dapat dilakuakn dengan mekanisme *roulette wheel selection,* dimana setiap individu pada populasi akan dibagi ke dalam roda lotere dengan proporsi sesuai dengan nilai *fitness* sehingga solusi dengan nilai *fitness* tertinggi memiliki probabilitas yang lebih besar untuk terpilih
5. *Crossover,* mengkombinasikan dua induk untuk menghasilkan individu atau kromosom baru yang memiliki kemungkinan untuk menghasilkan solusi yang lebih baik. Parameter penting dalam *crossover* adalah probabilitas *crossover* (P_o) yang selanjutnya akan dibandingkan dengan bilangan random (0,1). Apabila $r \leq P_o$ maka induk yang terpilih sebelumnya akan digantikan oleh hasil *crossover*. Namun apabila $r > P_o$, maka dua keturunan akan secara sederhana dihasilkan dengan meng-*copy* induknya
6. Mutasi, parameter penting dalam mekanisme mutasi adalah probabilitas mutasi (P_m) yang menunjukkan jumlah kromosom yang akan dimutasi. Untuk permasalahan kontinu, mekanisme mutasi dapat dilakukan dengan cara membangkitkan bilangan random. Sedangkan untuk permasalahan

diskrit mutasi dapat dilakukan dengan mekanisme sebagai berikut (Santosa dan Willy, 2011):

- Flip atau Membalik, berikut ilustrasinya:
  1 [**2 3 4**] 5 6 ➔ 1 [**4 3 2**] 5 6
- Swap atau Menukar, berikut ilustrasinya:
  1 [**2** 3 4 **5**] 6 ➔ 1 [**5** 3 4 **2**] 6
- Slide atau Menggeser, berikut ilustrasinya:
  1 [2 3 **4** 5] 6 ➔ 1 [**4** 2 3 5] 6

2.7 Posisi Penelitian

Setelah dilakukan tinjauan pustaka baik melalui buku, artikel, maupun jurnal, selanjutnya akan dijelaskan mengenai posisi penelitian tugas akhir. Sebelumnya, telah dilakukan penelitian terhadap permasalahan *single stage capacitated warehouse location problem* menggunakan algoritma *simulated annealing* oleh Kresna (2014) dengan waktu komputasi yang cukup kompetitif namun hasil solusi yang dihasilkan cukup jauh dari optimal. Penelitian ini menawarkan penyelesaian permasalahan *single stage capacitated warehouse location problem* dengan menggunakan algoritma *hybrid cross entropy* dan *genetic algorithm* dengan tujuan mampu mendapatkan hasil solusi yang lebih optimal. Penelitian-penelitian mengenai *single stage capacitated warehouse location problem* dan penelitian mengenai *hybrid cross entropy – genetic algorithm* yang telah dilakukan sebelumnya ditunjukkan pada Tabel 2.1 berikut.

Tabel 2. 1 Posisi Penelitian

No	Penelitian	Metode	Objektif	Hasil
1	(Verma et al., 2011) *Vertical Decomposition Approach to Solve Single Stage Capacitated Warehouse Location Problem*	*Vertical Decomposition*	Minimasi total biaya *fixed cost* dan *varibel cost* yang terdiri dari biaya distribusi dan biaya pendirian gudang	Dibuktikan bahwa metode *vertical decomposition mampu* SSCWLP yang lebih baik dibandingkan penelitian sebelumnya menggunakan *lagrange relaxation* oleh (Sharma et al. 2007)
2	(Yang et al., 2012) *A Cut and Solve Based Algorithm for the Single Source Capacitated Facility Location Problem*	*Cut and Solve Algorithm*	Minimasi total biaya pendirian fasilitas dan menempatkan konsumen untuk masing-masing fasilitas; Mengembangkan algoritma *cut and solve*	*cut and solve algorithm* yang merupakan penggabungan antara *cutting plane* method dan partial integrality dapat menyelesaikan permasalahan SSCFLP dan mampu mengurangi *optimality gap* dengan solusi optimal
3	(Guastaroba, et al. 2014) *A Heuristic for BILP Problems: The Single Source Capacitated Facility Location Problem*	*Kernel Search*	Minimasi biaya pendirian fasilitas dan *supply* konsumen	Metode *kernel search* cukup efektif untuk menyelesaikan SSCFLP namun dengan batasan bahwa setiap konsumen hanya dapat dilayani oleh satu fasilitas. Selain itu secara umum hasil yang didapatkan untuk 165 dari 170 sampel menunjukkan hasil yang cukup baik, namun untuk 5 sampel yang lain dihasilkan hasil yang error

Tabel 2. 1 Posisi Penelitian (Lanjutan)

No	Penelitian	Metode	Objektif	Hasil
4	(Rahmaniani, et al. 2015) *An Algorithm with Different Exploration Mechanism: Experimental Results to Capacitated Facility Location/Network Design Problems*	*Variable Neighborhood Search*	Mengembangkan algoritma yang secara efektif mampu menyelesaikan permasalahan FLND dengan mempertimbangkan batasan kapasitas fasilitas untuk permasalahan dengan ukuran *instance* yang realistis	Dikembangkan 3 metode dalam penyelesaian CFLND, yaitu *basic* VNS, *projection based* VNS, dan k-opt VNS; *Basic* VNS menunjukkan hasil yang buruk dalam penyelesaian CFLND, sedangkan hasil terbaik ditunjukkan oleh k-opt VNS dengan rata-rata *gap* dari nilai optimal sebesar 2.27%
5	(Ho, 2015) *An Iterated Tabu Search Heuristic for the Single Source Capacitated Facility Location Problem*	*Iterated Tabu Search*	Mengembangkan algoritma *iterated tabu search* untuk menyelesaikan permasalahan SSCFLP	Iterated tabu search dapat menghasilkan solusi optimal untuk 40 dari 57 *instance* yang dibandingkan. Selain itu, algoritma *iterated tabu search* ini juga menunjukkan hasil yang cukup kompetitif ketika dibandingkan dengan algoritma 22euristic yang lain seperti *repeated matching, very large-scale neighborhood, scatter search,* dan *hybrid ant colony – lagrange relaxation*

Tabel 2. 1 Posisi Penelitian (Lanjutan)

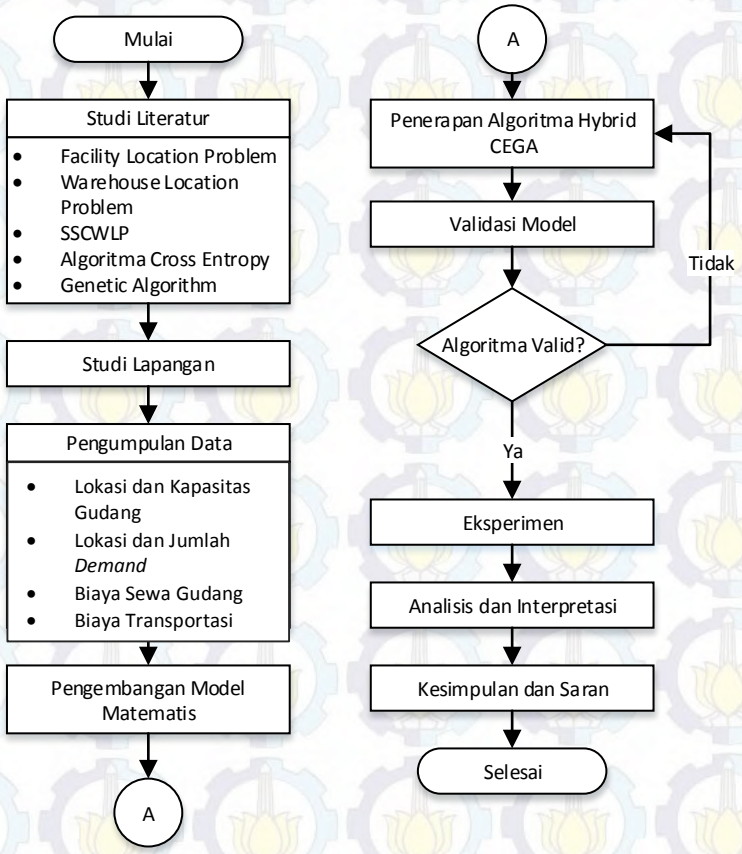
No	Penelitian	Metode	Objektif	Hasil
6	(Alizadeh et al., 2015) *A Capacitated Location-Allocation Problem with stochastic demands using sub-sources: An Empirical Study*	*Genetic Algorithm* dan *Colonial Competitive Algorithm*	Menentukan lokasi fasilitas dan alokasi konsumen yang optimal sehingga menghasilkan biaya yang minimum serta melakukan simplifikasi model untuk mengurangi waktu komputasi dalam menyelesaikan permasalahan SCLAPBDS	Secara umum algoritma GA dan CCA menunjukkan hasil yang near optimal dalam menyelesaikan permasalahan lokasi dan alokasi ini. Namun, secara umum algoritma CCA menunjukkan hasil yang lebih baik dibandingkan GA dari segi waktu komputasi dan akurasi hasil yang ditunjukkan
7	(Arostegui Jr. et al., 2006) *An Empirical Comparison of Tabu Search, Simulated Annealing, and Genetic Algorithms for Facilities Location Problems*	*Tabu Search, Simulated Annealing, dan Genetic Algortihms*	Membadingkan performansi relatif TS, SA, dan GA, untuk beberapa tipe FLP	Untuk *capacitated facility location problem* dan *multiple period facilities location problems (time limited performance)* solusi terbaik ditunjukkan oleh algoritma *tabu search*, sedangkan untuk permasalahan *multiple commodities facilities location problems (time limited performance), CFLP (solution limited performance), MPFLP (solution limited performance)*, dan *MFLP (solution limited performance)* hasil terbaik ditunjukkan oleh *genetic algorithm*

Tabel 2. 1 Posisi Penelitian (Lanjutan)

No	Penelitian	Metode	Objektif	Hasil
8	(Kresna, 2014) Algoritma S*imulated Annealing* untuk Menyelesaikan *Single Stage Capacitated Warehouse Location Problem* (Studi Kasus: PT. Petrokimia Gresik)	*Simulated Annealing*	Memberikan alternatif lokasi gudang penyangga pada PT. Petrokimia Gresik dengan biaya yang minimum	Algoritma *simulated annealing* yang dikembangkan dapat menyelesaikan permasalahan SSCWLP dengan waktu komputasi yang cepat namun solusi yang dihasilkan masih jauh dari optimal
9	(Santosa et al., 2014) *The Development of Hybrid Cross Entropy - Genetic Algortihm for Multi-Product Inventory Ship Routing Problem with Heterogeneous Fleet*	*Hybrid* CEGA	Meminimumkan biaya sistem yang terdiri dari biaya perjalanan, *port setup,* sewa kapal, dan biaya pencucian kompartemen	Algoritma *hybrid* CEGA mampu menyelesaikan permasalahan dengan waktu kompuasi yang cepat dan mampu menghasilkan solusi global optimal untuk kasus sederhana, serta lebih baik dibandingkan algoritma *hybrid tabu search*
10	Penelitian ini	*Hybrid* CEGA	Memberikan alternatif lokasi gudang penyangga pada PT. Petrokimia Gresik dengan biaya yang minimum	Diharapkan algortima *hybrid* CEGA yang dikembangkan dapat menyelesaikan permasalahan SSCWLP dengan solusi yang lebih mendekati optimal

BAB 3
METODOLOGI PENELITIAN

Bab ini menguraikan tahapan-tahapan penelitian yang dilakukan untuk menyelesaikan permasalahan *single stage capacitated warehouse location problem* dengan studi kasus di PT. Petrokimia Gresik dengan menggunakan algoritma *hybrid cross entropy* dan *genetic algorithm.* Secara garis besar, langkah-langkah penelitian yang digunakan penulis ditunjukkan pada *flowchart* pada Gambar 3.1 berikut.



Gambar 3. 1 *Flowchart* Metodologi Penelitian

3.1 Studi Literatur

Pada tahapan ini, hal yang dilakukan adalah melakukan studi literatur terhadap buku, artikel, dan jurnal-jurnal yang relevan untuk mendapatkan dasar teori dan referensi yang terkait dengan *single stage capacitated warehouse location problem,* algoritma *cross entropy,* dan *genetic algorithm.* Selain itu, studi literatur juga dilakukan terhadap penelitian terdahulu sehingga dapat dilakukan

pengembangan terhadap penelitian tersebut. Tahapan ini dilakukan dengan tujuan untuk menunjang jalannya penelitian sebagai sumber atau referensi yang dapat dijadikan dasar pemikiran.

3.2 Studi Lapangan

Tahapan studi lapangan dilakukan untuk mengetahui kondisi riil pada PT. Petrokimia Gresik serta melakukan validasi antara kondisi perusahaan pada penelitian terdahulu dengan kondisi riil perusahaan saat ini. Studi lapangan ini bertujuan untuk mengetahui proses distribusi pupuk pada daerah distribusi wilayah II PT. Petrokimia Gresik.

Dari proses studi lapangan diketahui bahwa proses distribusi pupuk wilayah II yang meliputi daerah di luar Pulau Jawa dan Bali dikategorikan menjadi empat lini. Lini 1 merupakan pabrik PT. Petrokimia Gresik yang berada di Gresik. Selanjutnya, pupuk yang diproduksi di lini 1 akan didistribusikan ke gudang lini 2 yang merupakan gudang penyangga yang terletak di ibu kota provinsi menggunakan kapal. Kemudian akan didistribusikan lagi ke gudang lini 3 yang merupakan gudang penyangga yang terletak di kabupaten/kota. Selanjutnya, lini 4 adalah distributor yang akan mengambil pupuk dari gudang lini 2 maupun gudang lini 3.

3.3 Pengumpulan Data

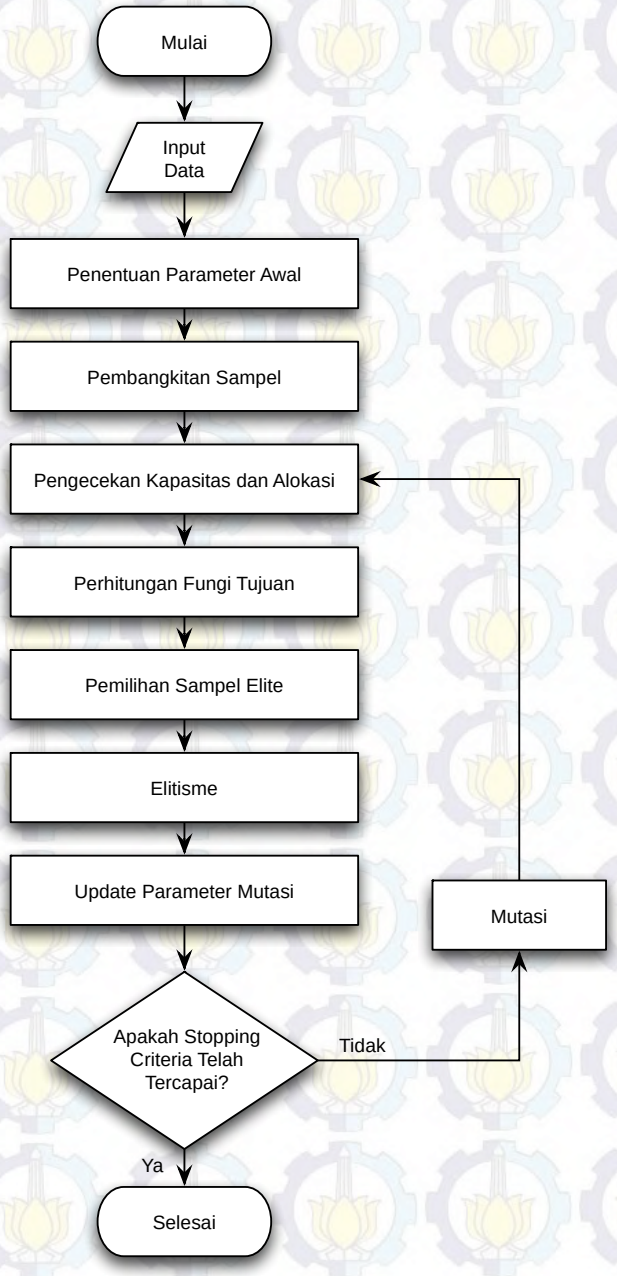
Tahapan selanjutnya merupakan pengumpulan data yang dibutuhkan untuk menyelesaikan *single stage capacitated warehouse location problem.* Data yang dibutuhkan antara lain lokasi *warehouse* lini 2, lokasi *warehouse* lini 3, kapasitas *warehouse* lini 2 dan lini 3, biaya sewa *warehouse,* biaya transportasi, lokasi *demand,* dan jumlah *demand.*

Biaya transportasi dibedakan menjadi dua, yaitu biaya transportasi dari gudang lini 2 ke gudang lini 3 dan biaya transportasi menuju lokasi *demand*. Untuk biaya transportasi dari gudang lini 2 ke gudang lini 3 didapatkan dari dokumen PT. Petrokimia Gresik yang berisikan perjanjian transportasi pupuk dengan pihak ketiga. Sedangkan biaya transportasi menuju lokasi *demand* didapatkan dengan cara menjari jarak antar lokasi dikalikan dengan biaya

transportasi per ton per kilometer yang disesuaikan dengan moda transportasi yang digunakan.

3.4 Penerapan Algoritma *Hybrid Cross Entropy – Genetic Algorithm*

Tahapan selanjutnya adalah penerapan algoritma *hybrid* CEGA untuk menyelesaikan *single stage capacitated warehouse location problem.* Penjelasan langkah-langkah algoritma *hybrid* CEGA ditunjukkan pada *flowchart* pada Gambar 3.2 berikut.



Gambar 3. 2 *Flowchart* Algoritma *Hybrid* CEGA

1. *Input* Data

   Input data yang diperlukan antara lain sebagai berikut:

   - Matriks jarak antara lokasi *warehouse* lini 2, *warehouse* lini 3, dan *demand*
   - Jumlah *demand* tiap lokasi
   - Kapasitas *warehouse* lini 2 dan lini 3
   - Komponen biaya *warehouse*
   - Biaya transportasi dari *warehouse* lini 2 ke *warehouse* lini 3 dan biaya transportasi per ton per kilometer menuju lokasi *demand*

2. Menentukan Parameter Awal

   Beberapa parameter yang ditentukan antara lain:

   - N = jumlah sampel
   - ρ = proporsi sampel elite
   - α = koefisien *smoothing*
   - P_m = probabilitas mutasi
   - ε = kriteria pemberhentian

3. Pembangkitan sampel

   Sampel yang dibangkitkan pada iterasi pertama merupakan sampel yang dibangkitkan secara *random.* Sedangkan pada pembangkitan sampel iterasi berikutnya dilakukan menggunakan mekanisme mutasi pada *genetic algorithm* menggunakan salah satu dari tiga mekanisme yaitu *swap, flip,* atau *slide.*

4. Pengecekan Kapasitas dan Alokasi

   Pada tahap ini akan dilakukan pengecekan apakah *demand* yang dilayani melebih kapasitas gudang lini 3 atau tidak. Apabila melebihi kapasitas maka *demand* akan dipindahkan ke gudang lini 3 yang lain atau di-*supply* langsung melalui gudang lini 2. Sedangkan gudang lini 3 yang tidak melayani *demand* akan dihapus. Mekanisme pengecekan kapasitas dan alokasi yang digunakan dalam tahapan ini diadaptasi dari penelitian yang telah dilakukan sebelumnya (Kresna, 2014).

5. Perhitungan Fungsi Tujuan

   Nilai fungsi tujuan dihitung menggunakan fungsi tujuan yang ingin dicapai yaitu total biaya transportasi dan biaya sewa *warehouse.*

6. Pemilihan Sampel Elite

   Dilakukan seleksi populasi untuk mendapatkan sampel elite sejumlah $\rho*N$ dari jumlah sampel yang diurutkan berdasarkan sampel yang telah diurutkan dari sampel dengan nilai fungsi tujuan terkecil hingga terbesar.

7. Elitisme

   Elitisme dilakukan dengan menyimpan sampel yang memiliki nilai terbaik sesuai dengan sampel yang telah terpilih pada tahap 6.

8. Update Parameter Mutasi

   Sampel elite digunakan untuk melakukan *update* parameter mutasi (P_m). mekanisme *update* parameter mutasi dilakukan menggunakan formulasi (3.1) berikut.

   $$P_{m(it)} = \frac{A_{(it)}}{2} \quad (3.1)$$

   Dengan nilai $A_{(it)}$ didapatkan dari persamaan (3.2) dan (3.3) berikut.

   $$A_{(it)} = (1-\alpha)u + A_{(it-1)}\alpha \quad (3.2)$$

   $$u = \frac{\overline{z_e}}{2\, z_{best}} \quad (3.3)$$

   Dimana z_e merupakan rata-rata fungsi tujuan sampel elite dan z_{best} merupakan solusi terbaik pada tiap iterasi.

9. Pengecekan Kriteria Pemberhentian Iterasi

   Kriteria pemberhentian yang digunakan pada algoritma ini adalah jumlah maksimum iterasi dan nilai toleransi pemberhentian (ε).

   $$\varepsilon = |P_{m(it)} - P_{m(it-1)}| \quad (3.4)$$

   iterasi akan tercapai apabila salah satu kriteria pemberhentian telah tercapai dan solusi terbaik yang dicapai pada iterasi tersebut merupakan hasil akhir komputasi yang didapat. Namun, apabila kriteria pemberhentian belum tercapai, maka akan dilakukan mekanisme mutasi.

10. Mutasi

    Mutasi dilakukan sebagai mekanisme untuk keluar dari jebakan lokal optimal. Mekanisme ini dilakukan dengan cara *flip, swap,* atau *slide* yang

dipilih dengan membangkitkan bilangan *random*. Apabila bilangan random yang dibangkitkan bernilai $r < 0.33$, maka dilakukan mutasi dengan teknik *flip.* Apabila bilangan *random* yang dibangkitkan bernilai antara $0.33 < r < 0.67$, maka akan dilakukan mutasi dengan teknik *swap.* Dan apabila nilai bilangan *random* yang dibangkitkan $r > 0.67$, maka akan dilakukan mutasi dengan teknik *slide.*

3.5 Validasi Algoritma

Validasi algoritma dilakukan untuk memastikan bahwa algoritma yang dikembangkan telah sesuai dengan permasalahan *single stage capacitated warehouse location problem.* Pada penelitian ini validasi dilakukan dengan cara memeriksa hasil eksperimen dengan perhitungan enumerasi pada sampel kecil untuk kemudian dibandingkan dengan hasil algoritma yang dilakukan secara manual sesuai *flowchart* pada Gambar 3.2.

3.6 Eksperimen

Tahap eksperimen dilakukan dengan cara melakukan *running* beberapa kali dengan mengubah beberapa parameter yang digunakan dalam algoritma ini untuk mendapatkan hasil terbaik. Selanjutnya hasil komputasi dibandingkan dengan metode lain untuk mengetahui performansi algoritma *hybrid cross entropy – genetic algorithm.* Pembanding yang digunakan adalah metode *simulated annealing, cross entropy, genetic algorithm* dan metode eksak.

3.7 Analisis dan Interpretasi

Pada tahap analisis dan interpretasi dilakukan analisis terkait pengaruh penetapan kombinasi parameter awal *hybrid cross entropy – genetic algorithm* sehingga diperoleh kombinasi parameter awal yang baik untuk data uji. Selain itu juga dilakukan analisis perbandingan performansi hasil algoritma *hybrid cross entropy – genetic algorithm* terhadap algoritma *simulated annealing, cross entropy,* dan *genetic algorithm* serta terhadap metode eksak yang akan ditinjau dari solusi yang dihasilkan serta waktu komputasi dalam menyelesaikan permasalahan

3.8 Kesimpulan dan Saran

Setelah tahap analisis dan interpretasi dilakukan, selanjutnya dilakukan penarikan kesimpulan terkait hasil eksperimen yang telah dilakukan. Setelah itu akan diberikan saran-saran yang dapat dijadikan sebagai rekomendasi untuk acuan penelitian selanjutnya.

BAB 4
PENGEMBANGAN MODEL DAN ALGORITMA

Pada bab pengembangan model dan algoritma akan dijelaskan mengenai tahapan-tahapan yang dilakukan dalam mengembangkan model *single stage capacitated warehouse location problem* dan algoritma *hybrid cross entropy genetic algorithm* yang digunakan untuk menyelesaikan permasalahan.

4.1 Pengembangan Model *Single Stage Capacitated Warehouse Location Problem*

Model *single stage capacitated warehouse location problem* digambarkan dalam bentuk model matematis yang terdiri dari beberapa bagian yakni parameter dan variabel keputusan yang digunakan dalam model, fungsi tujuan yang ingin dicapai, serta batasan-batasan dalam model.

4.1.1 Notasi dan Parameter

Berikut ini merupakan parameter-parameter yang terdapat dalam model matematis *single stage capacitated warehouse location problem,* antara lain:

i = gudang lini 2

j = gudang lini 3

k = *customer*

cap_j = kapasitas gudang lini 3 ke-j

c_{ij} = biaya pengiriman per ton dari gudang lini 2 ke-i menuju gudang lini 3 ke-j

ul_j = biaya *unloading* per ton di gudang lini 3 ke-j

d_{ik} = jarak gudang lini 2 ke-i menuju *customer* ke-k

d_{jk} = jarak gudang lini 3 ke-j menuju *customer* ke-k

4.1.2 Variabel Keputusan

Berikut ini merupakan variabel-variabel keputusan yang digunakan dalam model matematis *single stage capacitated warehouse location problem,* antara lain:

y_j = variabel biner yang menunjukkan keputusan membuka gudang lini 3 ke-*j*, bernilai 1 apabila gudang lini 3 *j* didirikan dan bernilai 0 apabila gudang lini 3 *j* tidak didirikan

x_{ij} = jumlah pupuk yang dikirim dari gudang lini 2 ke-*i* menuju gudang lini 3 ke-*j*

x_{ik} = jumlah pupuk yang dikirim dari gudang lini 2 ke-*i* menuju *customer* ke-*k*

x_{jk} = jumlah pupuk yang dikirim dari gudang lini 3 ke-j menuju *customer* ke-*k*

bin_{ik} = variabel biner yang menunjukkan alokasi pengiriman dari gudang lini 2 ke-*i* menuju *customer* ke-*k*, bernilai 1 apabila *customer k* dipasok oleh gudang lini 2 *i* dan bernilai 0 apabila tidak ada pasokan pupuk menuju *customer k* dari gudang lini 2 *i*

bin_{jk} = variabel biner yang menunjukkan alokasi pengiriman dari gudang lini 3 ke-*j* menuju *customer* ke-*k*, bernilai 1 apabila *customer k* dipasok oleh gudang lini 3 *j* dan bernilai 0 apabila tidak ada pasokan pupuk menuju *customer k* dari gudang lini 3 *j*

4.1.3 Fungsi Tujuan

Berikut merupakan fungsi tujuan yang dipertimbangkan dalam penyelesaian *single stage capacitated warehouse location problem*. Fungsi tujuan dalam permasalahan ini adalah meminimasi total biaya distribusi pupuk di Sumatera yang terdiri dari lima komponen biaya yang meliputi biaya sewa gudang, biaya transportasi, dan biaya *unloading* atau bongkar muat.

$$min \ \sum_i \sum_j x_{ij} c_{ij} + \sum_i \sum_k x_{ik} d_{ik} b + \sum_j \sum_k x_{jk} d_{jk} b + \sum_j f_j y_j + \sum_j ul_j \sum_i x_{ij} \quad (4.1)$$

Berdasarkan persamaan 4.1, akan dijelaskan mengenai masing-masing komponen biaya yang terdapat dalam fungsi tujuan 4.1 sebagai berikut, yaitu:

- $\sum_i \sum_j x_{ij} c_{ij}$

  Komponen biaya yang pertama adalah total biaya pengiriman dari gudang lini 2 menuju gudang lini 3. Biaya pengiriman ini didapatkan dari perkalian antara jumlah pupuk (ton) yang dikirimkan dari gudang

lini 2 ke-*i* menuju gudang lini 3 ke-*j* dengan biaya pengiriman per ton yang didapatkan dari dokumen kerjasama antara PT Petrokimia Gresik dengan pihak ketiga penyedia jasa transportasi.

- $\sum_i \sum_k x_{ik} d_{ik} b$

  Komponen biaya yang kedua adalah total biaya pengiriman dari gudang lini 2 menuju lokasi *demand*. Biaya ini didapatkan dari perkalian antara jumlah pupuk (ton) yang dikirimkan dari gudang lini 2 ke-*i* menuju *customer* ke-*k* dengan biaya pengiriman per ton, dimana biaya pengiriman per ton didapatkan dari pendekatan *bbm rate* per ton per km dikalikan dengan jarak tempuh dari lokasi gudang lini 2 ke-*i* menuju lokasi *demand* atau *customer* ke-*k*.

- $\sum_j \sum_k x_{jk} d_{jk} b$

  Komponen biaya yang ketiga adalah total biaya pengiriman dari gudang lini 3 menuju lokasi *demand*. Biaya ini didapatkan dari perkalian antara jumlah pupuk (ton) yang dikirimkan dari gudang lini 3 ke-*j* menuju *customer* ke-*k* dengan biaya pengiriman per ton, dimana biaya pengiriman per ton didapatkan dari pendekatan *bbm rate* per ton per km dikalikan dengan jarak tempuh dari lokasi gudang lini 3 ke-*j* menuju lokasi *demand* atau *customer* ke-*k*.

- $\sum_j f_j y_j$

  Komponen biaya yang keempat adalah total biaya fasilitas gudang lini 3 yang didirikan. Komponen biaya fasilitas f_i terdiri dari komponen biaya sewa gudang dan biaya pengelolaan stok yang merupakan biaya yang dibayarkan mingguan dan bernilai tetap, tidak dipengaruhi oleh banyaknya pupuk yang disimpan dalam gudang melainkan hanya ditentukan oleh keputusan membuka gudang lini 3 ke-*j*.

- $\sum_j ul_j \sum_i x_{ij}$

  Komponen biaya yang kelima adalah total biaya *unloading* atau biaya bongkar muat. Biaya ini didapatkan dari perkalian antara biaya *unloading* per ton di gudang lini 3 *j* dengan total pupuk (ton) yang dikirimkan menuju gudang lini 3 *j*.

4.1.4 Batasan

Berikut merupakan batasan-batasan yang dipertimbangkan dalam penyelesaian *single stage capacitated warehouse location problem*, antara lain:

- Batasan *Supply*

  $\sum_j x_{ij} + \sum_k x_{ik} \leq s_i \quad , \forall i$ (4.2)

  Persamaan 4.2 menujukkan bahwa jumlah pupuk yang dikirimkan dari gudang lini 2 ke-*i* menuju gudang lini 3 dan lokasi *demand* tidak melebihi kapasitas gudang lini 2 ke-*i* untuk semua *i.*

- Batasan Kapasitas Gudang Lini 3

  $\sum_i x_{ij} \leq y_j \, cap_j \quad , \forall j$ (4.3)

  Persamaan 4.3 menunjukkan bahwa jumlah pupuk yang dikirimkan menuju gudang lini 3 ke-*j* tidak melebihi kapasitas gudang *j* dan gudang *j* hanya bisa menerima pupuk ketika gudang lini 3 ke-*j* diputuskan untuk dibuka.

  $\sum_k x_{jk} \leq y_j \, cap_j \quad , \forall j$ (4.4)

  Persamaan 4.4 menunjukkan bahwa jumlah pupuk yang dikirimkan dari gudang lini 3 ke-*j* menuju lokasi *demand* tidak melebihi kapasitas gudang *j* dan gudang *j* hanya bisa mengirimkan pupuk ketika gudang lini 3 ke-*j* diputuskan untuk dibuka.

- Batasan Kesetimbangan *inflow* dan *outflow* Gudang Lini 3

  $\sum_i x_{ij} = \sum_k x_{jk} \quad , \forall j$ (4.5)

  Persamaan 4.5 memastikan bahwa jumlah pupuk yang masuk ke gudang lini 3 ke-*j* sama dengan jumlah pupuk yang dikirimkan dari gudang lini 3 ke-*j* untuk setiap gudang lini 3 *j*. Batasan ini juga menjamin bahwa tidak ada persediaan di gudang lini 3 *j* dan memastikan bahwa terdapat kesesuaian antara aliran pupuk yang masuk ke gudang lini 3 dan keluar dari gudang lini 3.

- Batasan *Demand*

  $\sum_i x_{ik} + \sum_j x_{jk} \geq D_k \quad , \forall k$ (4.6)

  Persamaan 4.6 menunjukkan bahwa kebutuhan *demand* untuk setiap *customer k* terpenuhi baik melalui pengiriman dari gudang lini 2 maupun dari gudang lini 3.

- Batasan Alokasi *Demand*

  $x_{ik} \leq M\ bin_{ik}$ (4.7)

  Persamaan 4.7 menunjukkan keterkaitan antara variabel *real* x_{ik} dengan variabel biner bin_{ik}.

  $x_{jk} \leq M\ bin_{jk}$ (4.8)

  Persamaan 4.8 menunjukkan keterkaitan antara variabel *real* x_{jk} dengan variabel biner bin_{jk}.

  $\sum_i bin_{ik} + \sum_j bin_{jk} = 1 \quad , \forall k$ (4.9)

  Persamaan 4.9 menunjukkan bahwa setiap *customer* hanya dapat dilayani oleh satu gudang saja, baik gudang lini 2 maupun gudang lini 3.

- Batasan Non-Negatif

  $x_{ij} \geq 0$ (4.10)

  $x_{jk} \geq 0$ (4.11)

  $x_{ik} \geq 0$ (4.12)

  $bin_{ik} \in \{0,1\}$ (4.13)

  $bin_{jk} \in \{0,1\}$ (4.14)

  $y_j \in \{0,1\}$ (4.15)

  Persamaan 4.10 hingga persamaan 4.12 merupakan persamaan yang menunjukkan bahwa variabel-variabel tersebut merupakan bilangan *real* non-negatif. Sedangkan persamaan 4.13 hingga persamaan 4.15 merupaakan persamaan yang menunjukkan bahwa variabel bernilai biner {0,1}.

4.2 Pengembangan Algoritma *Hybrid CE – GA* untuk SSCWLP

Berikut merupakan langkah-langkah yang terdapat dalam algoritma *hybrid CE – GA* yang digunakan untuk menyelesaikan *single stage capacitated warehouse location problem*:

Langkah 1: Input Data

Berikut merupakan data-data yang digunakan untuk menguji algoritma dan model matematis yang digunakan, antara lain:

1. Kapasitas Gudang Lini 2

Tabel 4. 1 Kapasitas Gudang Lini 2

Indeks Gudang Lini 2	1	2
Kapasitas	40	40

2. Kapasitas Gudang Lini 3

Tabel 4. 2 Kapasitas Gudang Lini 3

Indeks Gudang Lini 3	1	2	3
Kapasitas	39	35	31

3. *Demand*

Tabel 4. 3 Jumlah *Demand*

Indeks *Customer*	1	2	3	4
Demand	15	17	22	12

4. Biaya Sewa Gudang Lini 3

Tabel 4. 4 Biaya Sewa Gudang Lini 3

Indeks Gudang Lini 3	**Biaya Sewa Gudang**
1	1,500,000
2	1,250,000
3	1,000,000

5. Biaya Pengiriman dari Gudang Lini 2 Menuju Gudang Lini 3

Tabel 4. 5 Biaya Pengiriman dari Gudang Lini 2 menuju Gudang Lini 3

GL III / GL II	**GL III - 1**	**GL III - 2**	**GL III - 3**
GL II - 1	50000	64000	73000
GL II - 2	44000	61000	49000

6. Jarak dari Gudang Lini 2 Menuju Lokasi *Demand*

Tabel 4. 6 Jarak dari Gudang Lini 2 Menuju Lokasi *Demand*

***Cust* / GL II**	**C -1**	**C - 2**	**C - 3**	**C - 4**
GL II - 1	500	1250	1100	950
GL II - 2	1050	950	850	1100

7. Jarak dari Gudang Lini 3 Menuju Lokasi *Demand*

Tabel 4. 7 Jarak Gudang Lini 3 - *Customer*

***Cust* / GL III**	**C -1**	**C - 2**	**C - 3**	**C - 4**
GL III - 1	120	150	175	200
GL III - 2	140	170	125	110
GL III - 3	170	180	125	115

8. Biaya Transportasi per Ton per KM

   Dalam contoh kasus yang digunakan, biaya transportasi per ton per kilometer sebesar 8500.

Langkah 2: Inisialisasi Parameter

Parameter awal algoritma *hybrid* CE – GA yang digunakan dalam contoh kasus ini antara lain:

- Ukuran sampel (N) = 3
- Ukuran sampel elite (ρ) = 0.3
- Parameter *smoothing* (α) = 0.8
- Probabilitas Mutasi (P_m) = 1
- Kriteria Pemberhentian (ε) = 0.0001

Langkah 3: Pembangkitan Sampel

Pembangkitan sampel dilakukan dengan cara membangkitkan urutan bilangan secara random sebagai urutan lokasi gudang lini 2, gudang lini 3, dan *customer* yang secara berurutan diwakili oleh x1, x2, dan x3 seperti pada Tabel 4.8 berikut.

Tabel 4. 8 Pembangkitan Bilangan Random

Individu	X1	X2	X3
1	1 - 2	1 - 3 - 2	4 - 3 - 1 - 2
2	1 - 2	1 - 3 - 2	4 - 3 - 1 - 2
3	1 - 2	1 - 2 - 3	2 - 4 - 3 - 1

Selanjutnya dilakukan pembentukan matriks yang menunjukkan identitas fasilitas, nama fasilitas, dan kapasitas terpasang maupun *demand* fasilitas yang ditunjukkan pada Tabel 4.9 berikut.

Tabel 4. 9 Urutan Tiap Fasilitas

Gudang Lini 2		Gudang Lini 3			Customer			
1	1	2	2	2	3	3	3	3
1	2	1	3	2	4	3	1	2
40	40	39	31	35	12	22	15	17
1	1	2	2	2	3	3	3	3
1	2	1	3	2	4	3	1	2
40	40	39	31	35	12	22	15	17
1	1	2	2	2	3	3	3	3
1	2	1	2	3	2	4	3	1
40	40	39	35	31	17	12	22	15

Matriks yang terbentuk pada Tabel 4.9 terdiri dari 3 baris untuk setiap individu yang dibangkitkan. Baris pertama dari matriks menunjukkan indeks jenis fasilitas yang dimaksud, dimana indeks 1 berarti gudang lini 2, indeks 2 berarti gudang lini 3, dan indeks 3 berarti *customer*. Selanjutnya dilakukan penggabungan antara urutan gudang lini 2 dengan urutan *customer* seperti pada Tabel 4.10 berikut.

Tabel 4. 10 Urutan Gudang Lini 2 dan *Customer*

1	3	3	1	3	3
1	4	3	2	1	2
40	12	22	40	15	17
1	3	3	1	3	3
1	4	3	2	1	2
40	12	22	40	15	17
1	3	3	1	3	3
1	2	4	2	3	1
40	17	12	40	22	15

Setelah itu disisipkan lokasi gudang lini 3 ke dalam matriks yang telah terbentuk pada Tabel 4.10 hingga terbentuk matriks struktur solusi akhir seperti yang ditunjukkan pada Tabel 4.11 berikut.

Tabel 4. 11 Struktur Solusi

1	2	3	2	3	1	2	3	3
1	1	4	3	3	2	2	1	2
40	39	12	31	22	40	35	15	17
1	2	2	3	2	3	1	3	3
1	1	3	4	2	3	2	1	2
40	39	31	12	35	22	40	15	17
1	3	2	3	1	2	3	2	3
1	2	1	4	2	2	3	3	1
40	17	39	12	40	35	22	31	15

Langkah 4: Pengecekan Kapasitas dan Alokasi

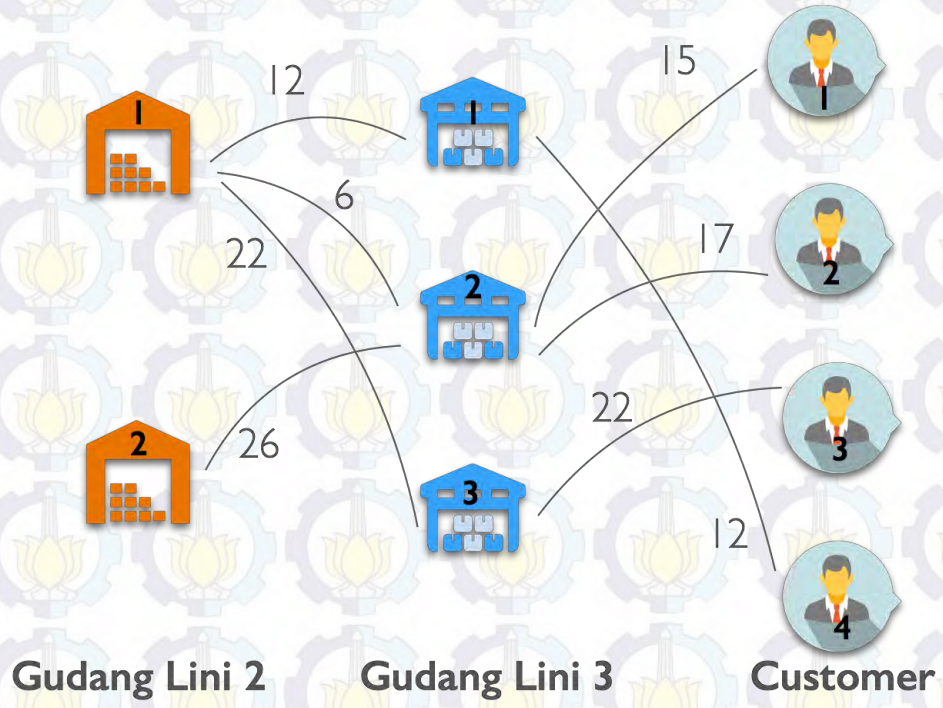
Berdasarkan struktur solusi yang didapatkan pada Tabel 4.11, selanjutnya dilakukan pengecekan kapasitas untuk memastikan bahwa *demand* yang dilayani oleh gudang lini 2 maupun gudang lini 3 tidak melebihi kapasitas gudang. Apabila melebihi kapasitas gudang, maka kelebihan *demand* tersebut akan dilayani melalui gudang lini 2 maupun gudang lini 3 yang lain. Setelah memastikan bahwa tidak ada kapasitas yang dilanggar, selanjutnya dilakukan perhitungan alokasi suplai dari gudang lini 2. Penentuan alokasi ini dilakukan dengan cara menarik

mundur dari permintaan *customer* menuju gudang lini 3 maupun gudang lini 2 yang melayani permintaan *customer* tersebut. *Output* dari perhitungan alokasi ini ditunjukkan pada Tabel 4.12 berikut.

Tabel 4. 12 Hasil Perhitungan Alokasi

individu 1	**GL III - 1**	**GL III - 2**	**GL III - 3**	**C - 1**	**C - 2**	**C - 3**	**C - 4**
GL II - 1	12	6	22	0	0	0	0
GL II - 2	0	26	0	0	0	0	0
GL III - 1	0	0	0	0	0	0	12
GL III - 2	0	0	0	15	17	0	0
GL III - 3	0	0	0	0	0	22	0
individu 2	**GL III - 1**	**GL III - 2**	**GL III - 3**	**C - 1**	**C - 2**	**C - 3**	**C - 4**
GL II - 1	0	13	12	15	0	0	0
GL II - 2	0	9	0	0	17	0	0
GL III - 1	0	0	0	0	0	0	0
GL III - 2	0	0	0	0	0	22	0
GL III - 3	0	0	0	0	0	0	12
individu 3	**GL III - 1**	**GL III - 2**	**GL III - 3**	**C - 1**	**C - 2**	**C - 3**	**C - 4**
GL II - 1	0	12	11	0	17	0	0
GL II - 2	15	0	11	0	0	0	0
GL III - 1	0	0	0	15	0	0	0
GL III - 2	0	0	0	0	0	0	12
GL III - 3	0	0	0	0	0	22	0

Gambaran alokasi distribusi pupuk hasil komputasi iterasi pertama untuk individu 1 sesuai hasil komputasi Tabel 4.12 ditunjukkan pada Gambar 4.1 berikut.



Gambar 4. 1 Alokasi Distribusi

Langkah 5: Perhitungan Fungsi Tujuan

Selanjutnya dilakukan perhitungan fungsi tujuan menggunakan persamaan 4.1 yang merupakan total biaya distribusi yang terdiri dari komponen biaya transportasi, biaya fasilitas, dan biaya *unloading*. Hasil perhitungan nilai fungsi tujuan untuk setiap individu ditunjukkan pada Tabel 4.13 berikut.

Tabel 4. 13 Total Biaya

Individu	Total Biaya
1	94,411,600
2	240,637,000
3	237,040,000

Langkah 6: Pemilihan Sampel Elite

Sesuai dengan ukuran sampel elite yang telah ditetapkan, maka dibutuhkan sampel elite sebanyak 0.3 x 3 = 0.9 atau dibulatkan menjadi satu individu. Berdasarkan hasil perhitungan fungsi tujuan pada Tabel 4.13 maka individu yang menjadi sampel elite adalah individu dengan total biaya minimum yaitu individu 1 dengan total biaya sebesar 94.411.600.

Langkah 7: Update Parameter Mutasi

Selanjutnya dilakukan *update* parameter mutasi sesuai dengan persamaan 3.1, 3.2, dan 3.3 sebagai berikut

$$u = \frac{94.411.600}{2\ x\ 94.411.600} = 0.5$$

$$A_{(it)} = (1 - 0.8)\ x\ 0.5 + 2\ x\ 0.8 = 1.7$$

$$P_{m(it)} = \frac{1.7}{2} = 0.85$$

Sehingga didapatkan bahwa probabilitas mutasi untuk iterasi pertama bernilai 0.85.

Langkah 8: Pengecekan Kriteria Pemberhentian Iterasi

Kriteria pemberhentian iterasi yang digunakan dalam algoritma ini merupakan batas absolut selisih antara probabilitas mutasi pada iterasi saat ini dengan probabilitas mutasi pada iterasi sebelumnya. Apabila nilai absolut selisih probabilitas mutasi lebih kecil dibandingkan batas kriteria pemberhentian, maka dapat dikatakan bahwa kriteria pemberhentian tercapai. Namun, apabila nilai

absolut selisih probabilitas mutasi lebih besar dibandingkan batas kriteria pemberhentian, maka kriteria pemberhentian belum tercapai dan akan dilakukan perhitungan kembali untuk iterasi berikutnya. Pengecekan kritera pemberhentian iterasi ini dilakukan menggunakan persamaan 3.4 seperti berikut.

$$\varepsilon = |0.85 - 1| = 0.15$$

Karena nilai absolut yang dihasilkan masih lebih besar dibandingkan batas kriteria pemberhentian, maka dilakukan perhitungan kembali untuk iterasi berikutnya menggunakan mekanisme mutasi.

Langkah 9: Mutasi

Mutasi dilakukan sebagai mekanisme untuk keluar dari jebakan lokal optimal. Mekanisme ini dilakukan dengan cara *flip, swap,* atau *slide* yang dipilih dengan membangkitkan bilangan *random*. Apabila bilangan random yang dibangkitkan bernilai $r < 0.33$, maka dilakukan mutasi dengan teknik *flip.* Apabila bilangan *random* yang dibangkitkan bernilai antara $0.33 < r < 0.67$, maka akan dilakukan mutasi dengan teknik *swap.* Dan apabila nilai bilangan *random* yang dibangkitkan $r > 0.67$, maka akan dilakukan mutasi dengan teknik *slide.* Dalam perhitungan yang dilakukan penulis, mutasi dilakukan dengan cara mengacak urutan x_1, x_2, dan x_3 untuk individu selain individu elite, yaitu individu 2 dan individu 3. Mekanisme mutasi yang dilakukan dapat dilihat pada Tabel 4.14 berikut.

Tabel 4. 14 Mekanisme Mutasi

Mutasi urutan X1			
Individu	**Random**	**Mekanisme Mutasi**	**X1 (new)**
2	0.22	Flip	2 - 1
3	0.79	Slide	2 - 1
Mutasi urutan X2			
Individu	**Random**	**Mekanisme Mutasi**	**X2 (new)**
2	0.14	Flip	1 - 2 - 3
3	0.92	Slide	1 - 3 - 2
Mutasi urutan X3			
Individu	**Random**	**Mekanisme Mutasi**	**X3 (new)**
2	0.54	Swap	2 - 3 - 1 - 4
3	0.39	Swap	2 - 3 - 4 - 1

Selanjutnya akan dilakukan langkah 3 hingga langkah 9 kembali hingga kriteria pemberhentian tercapai.

4.3 Verifikasi dan Validasi

Verifikasi merupakan tahapan yang digunakan untuk mengetahui bahwa model yang digunakan tepat secara logis dan matematis. Sedangkan validasi merupakan tahapan yang dilakukan untuk mengetahui apalah model yang dibuat telah mampu merepresentasikan permasalahan yang akan diselesaikan oleh penulis. Dalam penelitian ini, validasi dan verifikasi dilakukan terhadap model matematis dan terhadap algoritma *hybrid cross entropy – genetic algorithm* yang digunakan.

4.3.1 Verifikasi dan Validasi Model Matematis

Verifikasi model dilakukan dengan cara mengecek kesesuaian antara model matematis yang telah dikembangkan pada subbab 4.1 dengan model yang di-*generate* dengan *script* LINGO pada Lampiran 2 menggunakan data skala kecil pada subbab 4.2. Model matematis yang *di-generate* dengan *script* LINGO adalah sebagai berikut:

```
MODEL:
 [_40] MIN= 4250000 * VOLUME_G_C_1_1 + 10625000 * VOLUME_G_C_1_2 +
9350000 *
  VOLUME_G_C_1_3 + 8075000 * VOLUME_G_C_1_4 + 8925000 *
VOLUME_G_C_2_1 + 8075000 *
  VOLUME_G_C_2_2 + 7225000 * VOLUME_G_C_2_3 + 9350000 *
VOLUME_G_C_2_4 + 1020000 *
  VOLUME_DC_C_1_1 + 1275000 * VOLUME_DC_C_1_2 + 1487500 *
VOLUME_DC_C_1_3 + 1700000
  * VOLUME_DC_C_1_4 + 1190000 * VOLUME_DC_C_2_1 + 1445000 *
VOLUME_DC_C_2_2 +
  1062500 * VOLUME_DC_C_2_3 + 935000 * VOLUME_DC_C_2_4 + 1445000 *
VOLUME_DC_C_3_1 +
  1530000 * VOLUME_DC_C_3_2 + 1062500 * VOLUME_DC_C_3_3 + 977500 *
VOLUME_DC_C_3_4 +
  50000 * VOLUME_G_DC_1_1 + 64000 * VOLUME_G_DC_1_2 + 73000 *
VOLUME_G_DC_1_3 +
  44000 * VOLUME_G_DC_2_1 + 61000 * VOLUME_G_DC_2_2 + 49000 *
VOLUME_G_DC_2_3 +
  1500000 * OPENDC_1 + 1250000 * OPENDC_2 + 1000000 * OPENDC_3;
 [_2] VOLUME_G_C_2_1 + VOLUME_G_C_2_2 + VOLUME_G_C_2_3 +
VOLUME_G_C_2_4 +
  VOLUME_G_DC_2_1 + VOLUME_G_DC_2_2 + VOLUME_G_DC_2_3 <= 40;
 [_3] VOLUME_G_C_1_1 + VOLUME_G_C_2_1 + VOLUME_DC_C_1_1 +
VOLUME_DC_C_2_1 +
  VOLUME_DC_C_3_1 >= 15;
 [_4] VOLUME_G_C_1_2 + VOLUME_G_C_2_2 + VOLUME_DC_C_1_2 +
VOLUME_DC_C_2_2 +
  VOLUME_DC_C_3_2 >= 17;
 [_5] VOLUME_G_C_1_3 + VOLUME_G_C_2_3 + VOLUME_DC_C_1_3 +
VOLUME_DC_C_2_3 +
  VOLUME_DC_C_3_3 >= 22;
```

```
 [_6] VOLUME_G_C_1_4 + VOLUME_G_C_2_4 + VOLUME_DC_C_1_4 +
VOLUME_DC_C_2_4 +
  VOLUME_DC_C_3_4 >= 12;
 [_7] VOLUME_G_DC_1_1 + VOLUME_G_DC_2_1 - 39 * OPENDC_1 <= 0;
 [_8] VOLUME_G_DC_1_2 + VOLUME_G_DC_2_2 - 35 * OPENDC_2 <= 0;
 [_9] VOLUME_G_DC_1_3 + VOLUME_G_DC_2_3 - 31 * OPENDC_3 <= 0;
 [_10] VOLUME_DC_C_1_1 + VOLUME_DC_C_1_2 + VOLUME_DC_C_1_3 +
VOLUME_DC_C_1_4 - 39 *
  OPENDC_1 <= 0;
 [_11] VOLUME_DC_C_2_1 + VOLUME_DC_C_2_2 + VOLUME_DC_C_2_3 +
VOLUME_DC_C_2_4 - 35 *
  OPENDC_2 <= 0;
 [_12] VOLUME_DC_C_3_1 + VOLUME_DC_C_3_2 + VOLUME_DC_C_3_3 +
VOLUME_DC_C_3_4 - 31 *
  OPENDC_3 <= 0;
 [_13] VOLUME_DC_C_1_1 + VOLUME_DC_C_1_2 + VOLUME_DC_C_1_3 +
VOLUME_DC_C_1_4 -
  VOLUME_G_DC_1_1 - VOLUME_G_DC_2_1 = 0;
 [_14] VOLUME_DC_C_2_1 + VOLUME_DC_C_2_2 + VOLUME_DC_C_2_3 +
VOLUME_DC_C_2_4 -
  VOLUME_G_DC_1_2 - VOLUME_G_DC_2_2 = 0;
 [_15] VOLUME_DC_C_3_1 + VOLUME_DC_C_3_2 + VOLUME_DC_C_3_3 +
VOLUME_DC_C_3_4 -
  VOLUME_G_DC_1_3 - VOLUME_G_DC_2_3 = 0;
 [_16] VOLUME_DC_C_1_1 - 99999999 * BIN_DC_C_1_1 <= 0;
 [_17] VOLUME_DC_C_1_2 - 99999999 * BIN_DC_C_1_2 <= 0;
 [_18] VOLUME_DC_C_1_3 - 99999999 * BIN_DC_C_1_3 <= 0;
 [_19] VOLUME_DC_C_1_4 - 99999999 * BIN_DC_C_1_4 <= 0;
 [_20] VOLUME_DC_C_2_1 - 99999999 * BIN_DC_C_2_1 <= 0;
 [_21] VOLUME_DC_C_2_2 - 99999999 * BIN_DC_C_2_2 <= 0;
 [_22] VOLUME_DC_C_2_3 - 99999999 * BIN_DC_C_2_3 <= 0;
 [_23] VOLUME_DC_C_2_4 - 99999999 * BIN_DC_C_2_4 <= 0;
 [_24] VOLUME_DC_C_3_1 - 99999999 * BIN_DC_C_3_1 <= 0;
 [_25] VOLUME_DC_C_3_2 - 99999999 * BIN_DC_C_3_2 <= 0;
 [_26] VOLUME_DC_C_3_3 - 99999999 * BIN_DC_C_3_3 <= 0;
 [_27] VOLUME_DC_C_3_4 - 99999999 * BIN_DC_C_3_4 <= 0;
 [_28] VOLUME_G_C_1_1 - 99999999 * BIN_G_C_1_1 <= 0;
 [_29] VOLUME_G_C_1_2 - 99999999 * BIN_G_C_1_2 <= 0;
 [_30] VOLUME_G_C_1_3 - 99999999 * BIN_G_C_1_3 <= 0;
 [_31] VOLUME_G_C_1_4 - 99999999 * BIN_G_C_1_4 <= 0;
 [_32] VOLUME_G_C_2_1 - 99999999 * BIN_G_C_2_1 <= 0;
 [_33] VOLUME_G_C_2_2 - 99999999 * BIN_G_C_2_2 <= 0;
 [_34] VOLUME_G_C_2_3 - 99999999 * BIN_G_C_2_3 <= 0;
 [_35] VOLUME_G_C_2_4 - 99999999 * BIN_G_C_2_4 <= 0;
 [_36] BIN_G_C_1_1 + BIN_G_C_2_1 + BIN_DC_C_1_1 + BIN_DC_C_2_1 +
BIN_DC_C_3_1 = 1;
 [_37] BIN_G_C_1_2 + BIN_G_C_2_2 + BIN_DC_C_1_2 + BIN_DC_C_2_2 +
BIN_DC_C_3_2 = 1;
 [_38] BIN_G_C_1_3 + BIN_G_C_2_3 + BIN_DC_C_1_3 + BIN_DC_C_2_3 +
BIN_DC_C_3_3 = 1;
 [_39] BIN_G_C_1_4 + BIN_G_C_2_4 + BIN_DC_C_1_4 + BIN_DC_C_2_4 +
BIN_DC_C_3_4 = 1;
 [_1] VOLUME_G_C_1_1 + VOLUME_G_C_1_2 + VOLUME_G_C_1_3 +
VOLUME_G_C_1_4 +
  VOLUME_G_DC_1_1 + VOLUME_G_DC_1_2 + VOLUME_G_DC_1_3 <= 40;
 @BIN( BIN_G_C_1_1); @BIN( BIN_G_C_1_2); @BIN( BIN_G_C_1_3);
 @BIN( BIN_G_C_1_4); @BIN( BIN_G_C_2_1); @BIN( BIN_G_C_2_2);
 @BIN( BIN_G_C_2_3); @BIN( BIN_G_C_2_4); @BIN( BIN_DC_C_1_1);
 @BIN( BIN_DC_C_1_2); @BIN( BIN_DC_C_1_3); @BIN( BIN_DC_C_1_4);
```

```
 @BIN( BIN_DC_C_2_1); @BIN( BIN_DC_C_2_2); @BIN( BIN_DC_C_2_3);
 @BIN( BIN_DC_C_2_4); @BIN( BIN_DC_C_3_1); @BIN( BIN_DC_C_3_2);
 @BIN( BIN_DC_C_3_3); @BIN( BIN_DC_C_3_4); @BIN( OPENDC_1); @BIN(
OPENDC_2);
 @BIN( OPENDC_3);
 END
```

Berdasarkan hasil tersebut, dapat dibuktikan bahwa model yang di-*generate* telah sesuai dengan model matematis yang dikembangkan. Selanjutnya, dilakukan validasi model untuk memastikan bahwa logika perhitungan sesuai dengan kondisi permasalahan yang ada dengan cara melakukan pengujian batasan. Berikut merupahan hasil *running* software LINGO yang dilakukan penulis, yaitu:

```
  Global optimal solution found.
  Objective value:                              0.7788000E+08
  Objective bound:                              0.7788000E+08
  Infeasibilities:                              0.000000
  Extended solver steps:                                   0
  Total solver iterations:                                98

  Model Class:                                         MILP

  Total variables:                       55
  Nonlinear variables:                    0
  Integer variables:                     23

  Total constraints:                     40
  Nonlinear constraints:                  0

  Total nonzeros:                       165
  Nonlinear nonzeros:                     0

                      Variable           Value        Reduced Cost
                    SUPPLY( 1)        40.00000            0.000000
                    SUPPLY( 2)        40.00000            0.000000
                    KAPASITAS( 1)     39.00000            0.000000
                    KAPASITAS( 2)     35.00000            0.000000
                    KAPASITAS( 3)     31.00000            0.000000
                    SEWA( 1)          1500000.            0.000000
                    SEWA( 2)          1250000.            0.000000
                    SEWA( 3)          1000000.            0.000000
                    ULCOST( 1)        0.000000            0.000000
                    ULCOST( 2)        0.000000            0.000000
                    ULCOST( 3)        0.000000            0.000000
                    OPENDC( 1)        1.000000            1500000.
                    OPENDC( 2)        1.000000            1250000.
                    OPENDC( 3)        0.000000            1000000.
                    MASUK( 1)         0.000000            0.000000
                    MASUK( 2)         0.000000            0.000000
                    MASUK( 3)         0.000000            0.000000
                    DEMAND( 1)        15.00000            0.000000
                    DEMAND( 2)        17.00000            0.000000
```

DEMAND(3)	22.00000	0.000000
DEMAND(4)	12.00000	0.000000
BIAYA_G_DC(1, 1)	50000.00	0.000000
BIAYA_G_DC(1, 2)	64000.00	0.000000
BIAYA_G_DC(1, 3)	73000.00	0.000000
BIAYA_G_DC(2, 1)	44000.00	0.000000
BIAYA_G_DC(2, 2)	61000.00	0.000000
BIAYA_G_DC(2, 3)	49000.00	0.000000
VOLUME_G_DC(1, 1)	0.000000	3000.000
VOLUME_G_DC(1, 2)	26.00000	0.000000
VOLUME_G_DC(1, 3)	0.000000	73000.00
VOLUME_G_DC(2, 1)	32.00000	0.000000
VOLUME_G_DC(2, 2)	8.000000	0.000000
VOLUME_G_DC(2, 3)	0.000000	52000.00
JARAK_DC_C(1, 1)	120.0000	0.000000
JARAK_DC_C(1, 2)	150.0000	0.000000
JARAK_DC_C(1, 3)	175.0000	0.000000
JARAK_DC_C(1, 4)	200.0000	0.000000
JARAK_DC_C(2, 1)	140.0000	0.000000
JARAK_DC_C(2, 2)	170.0000	0.000000
JARAK_DC_C(2, 3)	125.0000	0.000000
JARAK_DC_C(2, 4)	110.0000	0.000000
JARAK_DC_C(3, 1)	170.0000	0.000000
JARAK_DC_C(3, 2)	180.0000	0.000000
JARAK_DC_C(3, 3)	125.0000	0.000000
JARAK_DC_C(3, 4)	115.0000	0.000000
VOLUME_DC_C(1, 1)	15.00000	0.000000
VOLUME_DC_C(1, 2)	17.00000	0.000000
VOLUME_DC_C(1, 3)	0.000000	408000.0
VOLUME_DC_C(1, 4)	0.000000	748000.0
VOLUME_DC_C(2, 1)	0.000000	187000.0
VOLUME_DC_C(2, 2)	0.000000	187000.0
VOLUME_DC_C(2, 3)	22.00000	0.000000
VOLUME_DC_C(2, 4)	12.00000	0.000000
VOLUME_DC_C(3, 1)	0.000000	378000.0
VOLUME_DC_C(3, 2)	0.000000	208000.0
VOLUME_DC_C(3, 3)	0.000000	0.000000
VOLUME_DC_C(3, 4)	0.000000	0.000000
BIN_DC_C(1, 1)	1.000000	0.000000
BIN_DC_C(1, 2)	1.000000	0.000000
BIN_DC_C(1, 3)	0.000000	0.000000
BIN_DC_C(1, 4)	0.000000	0.000000
BIN_DC_C(2, 1)	0.000000	0.000000
BIN_DC_C(2, 2)	0.000000	0.000000
BIN_DC_C(2, 3)	1.000000	0.000000
BIN_DC_C(2, 4)	1.000000	0.000000
BIN_DC_C(3, 1)	0.000000	0.000000
BIN_DC_C(3, 2)	0.000000	0.000000
BIN_DC_C(3, 3)	0.000000	-0.6400000E+13
BIN_DC_C(3, 4)	0.000000	-0.2150000E+13
JARAK_G_C(1, 1)	500.0000	0.000000
JARAK_G_C(1, 2)	1250.000	0.000000
JARAK_G_C(1, 3)	1100.000	0.000000
JARAK_G_C(1, 4)	950.0000	0.000000
JARAK_G_C(2, 1)	1050.000	0.000000
JARAK_G_C(2, 2)	950.0000	0.000000
JARAK_G_C(2, 3)	850.0000	0.000000
JARAK_G_C(2, 4)	1100.000	0.000000
VOLUME_G_C(1, 1)	0.000000	3183000.

VOLUME_G_C(1, 2)	0.000000	9303000.
VOLUME_G_C(1, 3)	0.000000	8223500.
VOLUME_G_C(1, 4)	0.000000	7076000.
VOLUME_G_C(2, 1)	0.000000	7861000.
VOLUME_G_C(2, 2)	0.000000	6756000.
VOLUME_G_C(2, 3)	0.000000	6101500.
VOLUME_G_C(2, 4)	0.000000	8354000.
BIN_G_C(1, 1)	0.000000	0.000000
BIN_G_C(1, 2)	0.000000	0.000000
BIN_G_C(1, 3)	0.000000	0.000000
BIN_G_C(1, 4)	0.000000	0.000000
BIN_G_C(2, 1)	0.000000	0.000000
BIN_G_C(2, 2)	0.000000	0.000000
BIN_G_C(2, 3)	0.000000	0.000000
BIN_G_C(2, 4)	0.000000	0.000000

Row	Slack or Surplus	Dual Price
1	14.00000	0.000000
2	0.000000	3000.000
3	0.000000	-1067000.
4	0.000000	-1322000.
5	0.000000	-1126500.
6	0.000000	-999000.0
7	7.000000	0.000000
8	1.000000	0.000000
9	0.000000	0.000000
10	7.000000	0.000000
11	1.000000	0.000000
12	0.000000	0.000000
13	0.000000	47000.00
14	0.000000	64000.00
15	0.000000	0.000000
16	0.9999998E+08	0.000000
17	0.9999998E+08	0.000000
18	0.000000	0.000000
19	0.000000	0.000000
20	0.000000	0.000000
21	0.000000	0.000000
22	0.9999998E+08	0.000000
23	0.9999999E+08	0.000000
24	0.000000	0.000000
25	0.000000	0.000000
26	0.000000	64000.00
27	0.000000	21500.00
28	0.000000	0.000000
29	0.000000	0.000000
30	0.000000	0.000000
31	0.000000	0.000000
32	0.000000	0.000000
33	0.000000	0.000000
34	0.000000	0.000000
35	0.000000	0.000000
36	0.000000	0.000000
37	0.000000	0.000000
38	0.000000	0.000000
39	0.000000	0.000000
40	0.7788000E+08	-1.000000

Selanjutnya dilakukan pengujian batasan terhadap hasil *running software* LINGO sebagai berikut:

- Batasan *Supply*

1. `VOLUME_G_C_1_1 + VOLUME_G_C_1_2 + VOLUME_G_C_1_3 + VOLUME_G_C_1_4 + VOLUME_G_DC_1_1 + VOLUME_G_DC_1_2 + VOLUME_G_DC_1_3 <= 40`

   $0 + 0 + 0 + 0 + 0 + 26 + 0 <= 40$

2. `VOLUME_G_C_2_1 + VOLUME_G_C_2_2 + VOLUME_G_C_2_3 + VOLUME_G_C_2_4 + VOLUME_G_DC_2_1 + VOLUME_G_DC_2_2 + VOLUME_G_DC_2_3 <= 40`

   $0 + 0 + 0 + 0 + 32 + 8 + 0 <= 40$

- Batasan Kapasitas Gudang Lini 3

1. `VOLUME_G_DC_1_1 + VOLUME_G_DC_2_1 - 39 * OPENDC_1 <= 0;`

   $0 + 32 – 39 * 1 <= 0$

2. `VOLUME_G_DC_1_2 + VOLUME_G_DC_2_2 - 35 * OPENDC_2 <= 0;`

   $26 + 8 – 34 * 1 <= 0$

3. `VOLUME_G_DC_1_3 + VOLUME_G_DC_2_3 - 31 * OPENDC_3 <= 0;`

   $0 + 0 – 31 * 0 <= 0$

4. `VOLUME_DC_C_1_1 + VOLUME_DC_C_1_2 + VOLUME_DC_C_1_3 + VOLUME_DC_C_1_4 - 39 * OPENDC_1 <= 0;`

   $15 + 17 + 0 + 0 – 39 * 1 <= 0$

5. `VOLUME_DC_C_2_1 + VOLUME_DC_C_2_2 + VOLUME_DC_C_2_3 + VOLUME_DC_C_2_4 - 35 * OPENDC_2 <= 0;`

   $0 + 0 + 22 + 12 – 34 * 1 <= 0$

6. `VOLUME_DC_C_3_1 + VOLUME_DC_C_3_2 + VOLUME_DC_C_3_3 + VOLUME_DC_C_3_4 - 31 * OPENDC_3 <= 0;`

   $0 + 0 + 0 + 0 – 31 * 0 <= 0$

- Batasan Kesetimbangan *inflow* dan *outflow* Gudang Lini 3

1. `VOLUME_DC_C_1_1 + VOLUME_DC_C_1_2 + VOLUME_DC_C_1_3 + VOLUME_DC_C_1_4 - VOLUME_G_DC_1_1 - VOLUME_G_DC_2_1 = 0;`

   $15 + 17 + 0 + 0 - 0 - 32 = 0$
2. `VOLUME_DC_C_2_1 + VOLUME_DC_C_2_2 + VOLUME_DC_C_2_3 + VOLUME_DC_C_2_4 - VOLUME_G_DC_1_2 - VOLUME_G_DC_2_2 = 0;`

   $0 + 0 + 22 + 12 - 26 - 8 = 0$
3. `VOLUME_DC_C_3_1 + VOLUME_DC_C_3_2 + VOLUME_DC_C_3_3 + VOLUME_DC_C_3_4 - VOLUME_G_DC_1_3 - VOLUME_G_DC_2_3 = 0;`

   $0 + 0 + 0 - 0 - 0 = 0$

- Batasan *Demand*

1. `VOLUME_G_C_1_1 + VOLUME_G_C_2_1 + VOLUME_DC_C_1_1 + VOLUME_DC_C_2_1 + VOLUME_DC_C_3_1 >= 15;`

   $0 + 0 + 15 + 0 + 0 >= 15$
2. `VOLUME_G_C_1_2 + VOLUME_G_C_2_2 + VOLUME_DC_C_1_2 + VOLUME_DC_C_2_2 + VOLUME_DC_C_3_2 >= 17;`

   $0 + 0 + 17 + 0 + 0 >= 17$
3. `VOLUME_G_C_1_3 + VOLUME_G_C_2_3 + VOLUME_DC_C_1_3 + VOLUME_DC_C_2_3 + VOLUME_DC_C_3_3 >= 22;`

   $0 + 0 + 0 + 22 + 0 >= 22$
4. `VOLUME_G_C_1_4 + VOLUME_G_C_2_4 + VOLUME_DC_C_1_4 + VOLUME_DC_C_2_4 + VOLUME_DC_C_3_4 >= 12;`

   $0 + 0 + 0 + 12 + 0 >= 12$

- Batasan Alokasi *Demand*

1. `VOLUME_DC_C_1_1 - 99999999 * BIN_DC_C_1_1 <= 0;`

   $15 - 99999999 * 1 <= 0$
2. `VOLUME_DC_C_1_2 - 99999999 * BIN_DC_C_1_2 <= 0;`

   $17 - 99999999 * 1 <= 0$
3. `VOLUME_DC_C_1_3 - 99999999 * BIN_DC_C_1_3 <= 0;`

   $0 - 99999999 * 0 <= 0$
4. `VOLUME_DC_C_1_4 - 99999999 * BIN_DC_C_1_4 <= 0;`

0 – 99999999 * 0 <= 0

5. VOLUME_DC_C_2_1 - 99999999 * BIN_DC_C_2_1 <= 0;

   0 – 99999999 * 0 <= 0

6. VOLUME_DC_C_2_2 - 99999999 * BIN_DC_C_2_2 <= 0;

   0 – 99999999 * 0 <= 0

7. VOLUME_DC_C_2_3 - 99999999 * BIN_DC_C_2_3 <= 0;

   22 – 99999999 * 1 <= 0

8. VOLUME_DC_C_2_4 - 99999999 * BIN_DC_C_2_4 <= 0;

   12 – 99999999 * 1 <= 0

9. VOLUME_DC_C_3_1 - 99999999 * BIN_DC_C_3_1 <= 0;

   0 – 99999999 * 0 <= 0

10. VOLUME_DC_C_3_2 - 99999999 * BIN_DC_C_3_2 <= 0;

    0 – 99999999 * 0 <= 0

11. VOLUME_DC_C_3_3 - 99999999 * BIN_DC_C_3_3 <= 0;

    0 – 99999999 * 0 <= 0

12. VOLUME_DC_C_3_4 - 99999999 * BIN_DC_C_3_4 <= 0;

    0 – 99999999 * 0 <= 0

13. VOLUME_G_C_1_1 - 99999999 * BIN_G_C_1_1 <= 0;

    0 – 99999999 * 0 <= 0

14. VOLUME_G_C_1_2 - 99999999 * BIN_G_C_1_2 <= 0;

    0 – 99999999 * 0 <= 0

15. VOLUME_G_C_1_3 - 99999999 * BIN_G_C_1_3 <= 0;

    0 – 99999999 * 0 <= 0

16. VOLUME_G_C_1_4 - 99999999 * BIN_G_C_1_4 <= 0;

    0 – 99999999 * 0 <= 0

17. VOLUME_G_C_2_1 - 99999999 * BIN_G_C_2_1 <= 0;

    0 – 99999999 * 0 <= 0

18. VOLUME_G_C_2_2 - 99999999 * BIN_G_C_2_2 <= 0;

    0 – 99999999 * 0 <= 0

19. VOLUME_G_C_2_3 - 99999999 * BIN_G_C_2_3 <= 0;

    0 – 99999999 * 0 <= 0

20. VOLUME_G_C_2_4 - 99999999 * BIN_G_C_2_4 <= 0;

    0 – 99999999 * 0 <= 0

21. `BIN_G_C_1_1 + BIN_G_C_2_1 + BIN_DC_C_1_1 + BIN_DC_C_2_1 + BIN_DC_C_3_1 = 1;`

$$0 + 0 + 1 + 0 + 0 = 1$$

22. `BIN_G_C_1_2 + BIN_G_C_2_2 + BIN_DC_C_1_2 + BIN_DC_C_2_2 + BIN_DC_C_3_2 = 1;`

$$0 + 0 + 1 + 0 + 0 = 1$$

23. `BIN_G_C_1_3 + BIN_G_C_2_3 + BIN_DC_C_1_3 + BIN_DC_C_2_3 + BIN_DC_C_3_3 = 1;`

$$0 + 0 + 0 + 1 + 0 = 1$$

24. `BIN_G_C_1_4 + BIN_G_C_2_4 + BIN_DC_C_1_4 + BIN_DC_C_2_4 + BIN_DC_C_3_4 = 1;`

$$0 + 0 + 0 + 1 + 0 = 1$$

Berdasarkan hasil pengujian batasan yang dilakukan dapat dibuktikan bahwa logika perhitungan dalam model yang dikembangkan telah sesuai. Hal ini ditunjukkan dengan tidak adanya batasan yang dilanggar dalam model.

4.3.2 Verifikasi dan Validasi Algoritma *Hybrid Cross Entropy – Genetic Algorithm*

Verifikasi dilakukan dengan cara mengecek kesesuaian logika antara komputasi algoritma pada *software* MATLAB dengan tahapan pada gambar *flowchart* 3.2. Selain itu, verifikasi juga dapat dilakukan untuk mengetahui apakah terdapat *error* dalam komputasi menggunakan *software* MATLAB. Pada kode program MATLAB untuk algoritma pada Lampiran 3 dapat dilihat bahwa tahapan komputasi telah sesuai dengan *flowchart.* Selain itu, kode program juga dapat berjalan tanpa terjadi *error,* sehingga dapat disimpulkan bahwa algoritma telah terverifikasi.

Validasi algoritma dilakkan dengan cara membandingkan antara hasil komputasi algoritma *hybrid* CE – GA dengan hasil komputasi pada metode eksak. Berikut adalah hasil komputasi algoritma *hybrid* CE – GA.

```
>> [mincost, alokasi_min]=CEGA_SSCWLP

mincost =

    77880000


alokasi_min =

     0    26     0     0     0     0     0
    32     8     0     0     0     0     0
     0     0     0    15    17     0     0
     0     0     0     0     0    22    12
     0     0     0     0     0     0     0
```

Berdasarkan nilai *output* yang dihasilkan, dapat disimpulkan bahwa pada permasalahan yang sama algoritma *hybrid* CE – GA dapat menghasilkan solusi yang sama dengan hasil komputasi menggunakan metode eksak. Nilai *output* yang dihasilkan menunjukkan solusi yang sama baik dari nilai fungsi tujuan maupun alokasi distribusinya, sehingga dapat disimpulkan bahwa algoritma *hybrid* CE – GA yang dikembangkan untuk *single stage capacitated warehouse location problem* telah valid.

BAB 5
EKSPERIMEN DAN ANALISIS

Pada bab ini akan dijelaskan mengenai tahapan eksperimen yang dilakukan beserta penjabaran hasil eksperimennya. Selanjutnya akan dilakukan analisis terkait hasil eksperimen yang dilakukan, meliputi analisis kondisi eksisting, analisis eksperimen dengan metode eksak, analisis eksperimen dengan algoritma *hybrid* CE – GA, serta analisis perbandingan hasil algoritma *hybrid* CE – GA dengan kondisi eksisting, metode eksak, dan algoritma metaheuristik lain.

5.1 Pengumpulan Data

Data yang digunakan dalam penelitian ini merupakan data sekunder yang didapatkan dari Departemen Distribusi Wilayah II PT. Petrokimia Gresik. Data yang dibutuhkan untuk menyelesaikan *single stage capacitated warehouse location problem* antara lain adalah lokasi dan kapasitas gudang lini 2, lokasi dan kapasitas gudang lini 3, Biaya sewa gudang, biaya transportasi per ton dari gudang lini 2 menuju gudang lini 3, lokasi dan jumlah *demand*, jarak gudang lini 2 menuju lokasi *demand*, jarak gudang lini 3 menuju lokasi *demand,* serta biaya bahan bakar per ton per kilometer. Data yang digunakan oleh penulis secara lengkap ditampilkan pada Lampiran 1.

Biaya transportasi dibedakan menjadi dua, yaitu biaya transportasi dari gudang lini 2 ke gudang lini 3 dan biaya transportasi menuju lokasi *demand*. Untuk biaya transportasi dari gudang lini 2 ke gudang lini 3 didapatkan dari dokumen PT. Petrokimia Gresik yang berisikan perjanjian transportasi pupuk dengan pihak ketiga. Sedangkan biaya transportasi menuju lokasi *demand* didapatkan dengan cara menjari jarak antar lokasi dikalikan dengan biaya transportasi per ton per kilometer yang disesuaikan dengan moda transportasi yang digunakan. Jarak antar lokasi yang meliputi jarak dari gudang lini 2 menuju lokasi *demand* dan jarak dari gudang lini 3 menuju lokasi *demand* didapatkan dengan cara mencari jarak dari lokasi gudang ke pusat kabupaten/kota sebagai lokasi dari *end customer* atau lokasi *demand.*

Biaya transportasi per ton per kilometer didapatkan dari pendekatan konsumsi bahan bakar. Biaya transportasi per ton per kilometer yang digunakan

oleh penulis sebesar Rp 89,33 per ton per km. Nilai ini didapatkan dari perkalian antara konsumi BBM per kilometer (dalam liter) dengan harga BBM (solar) per liter dibagi dengan kapasitas moda transportasi yang digunakan yaitu 15 ton. Perhitungan biaya transportasi per ton per kilometer secara rinci sebagai berikut.

$$Biaya\ Transportasi\ (per\ ton\ per\ km) = \frac{Konsumsi\ BBM\ x\ Harga\ Solar}{Kapasitas\ Moda\ Transportasi}$$

$$Biaya\ Transportasi\ (per\ ton\ per\ km) = \frac{0.2\ x\ 6700}{15}$$

$$Biaya\ Transportasi\ (per\ ton\ per\ km) = 89.33$$

Biaya terkait gudang terdiri dari dua komponen biaya yaitu biaya yang bersifat variabel dan biaya yang bersifat tetap. Biaya yang bersifat variabel terdiri dari biaya *unloading* atau biaya bongkar muat yang berbanding lurus dengan jumlah pupuk (ton) yang dikirimkan menuju gudang lini 3. Sedangkan biaya yang bersifat tetap terdiri dari dua komponen biaya yaitu biaya sewa dan biaya tenaga kerja yang nilainya hanya dipengaruhi oleh keputusan membuka atau menutup gudang tanpa dipengaruhi banyaknya pupuk yang dikirimkan ke gudang tersebut.

Jumlah *demand* untuk setiap lokasi didapatkan dari data aktual pengambilan pupuk mingguan oleh tiap kabupaten/kota pada periode Januari hingga November 2015. Berdasarkan data yang didapat tersebut selanjutnya dipilih nilai maksimal pengambilan per minggu sebagai dasar acuan jumlah *demand* kabupaten/kota.

5.2 Kondisi Eksisting

Perhitungan biaya pada kondisi eksisting distribusi pupuk oleh PT. Petrokimia Gresik di Pulau Sumatera hanya mencakup gudang lini 2 dan gudang lini 3 saja, tidak mencakup pengiriman menuju *end customer*. Komponen biaya yang diperhitungkan antara lain adalah biaya pengiriman dari gudang lini 2 menuju gudang lini 3, biaya sewa dan biaya tenaga kerja gudang lini 3, serta biaya bongkar muat. Perhitungan biaya yang dilakukan didasarkan pada data eksisting yang terdapat pada Lampiran 1. Berikut merupakan rincian perhitungan biaya yang dilakukan, yaitu:

- Biaya Transportasi

  Perhitungan biaya transportasi dilakukan dengan cara mengalikan antara alokasi pupuk untuk setiap gudang lini 3 dengan biaya transportasi per ton dari gudang lini 2 menuju gudang lini 3 yang terletak dalam *coverage area*-nya. Perhitungan ini dilakukan berdasarkan data eksisting pada Lampiran 1. Hasil komputasi yang dilakukan penulis menunjukkan bahwa pada kondisi eksisting dibutuhkan biaya transportasi pupuk setiap minggunya sebesar Rp 383.245.579,00.

- Biaya Sewa Gudang dan Tenaga Kerja

  Komponen biaya ini memiliki nilai yang berbeda untuk setiap gudangnya. Pada kondisi eksisting, seluruh gudang lini 3 yang ada (29 gudang) digunakan semua. Oleh karena itu, total biaya sewa gudang dan tenaga kerja pada kondisi eksisting didapatkan dari jumlah biaya sewa gudang dan tenaga kerja untuk seluruh gudang lini 3. Berdasarkan perhitungan yang dilakukan, didapatkan total biaya sewa gudang dan biaya tenaga kerja setiap minggunya sebesar Rp 196.733.794,00

- Biaya Bongkar Muat

  Sama seperti biaya sewa gudang dan biaya tenaga kerja, komponen biaya bongkat muat juga memiliki nilai yang berbeda untuk setiap gudangnya. Perhitungan total biaya bongkar muat didapatkan dari perkalian antara alokasi pupuk untuk setiap gudang lini 3 dengan biaya bongkar muat per ton di setiap gudang. Berdasarkan perhitungan yang dilakukan penulis didapatkan bahwa biaya bongkar muat yang dibtuhkan sebesar Rp 768.232.819,00

- Total Biaya

  Setelah dilakukan perhitungan biaya untuk tiap komponen biaya distribusi pupuk, selanjutnya dapat dilakukan perhitungan total biaya distribusi pupuk PT. Petrokimia Gresik di Pulau Sumatera. Berdasarkan perhitungan yang telah dilakukan didapatkan total biaya distribusi mingguan sebesar Rp 1.348.212.192,00 dengan rincian biaya transportasi, biaya sewa gudang dan tenaga kerja, serta biaya bongkar

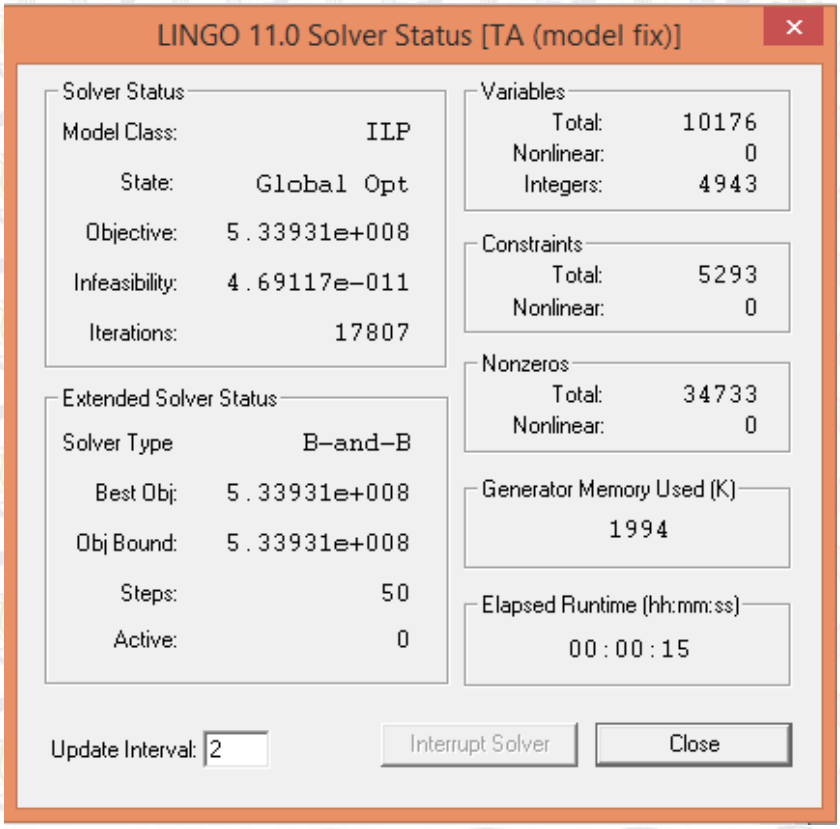
muat secara urut sebesar Rp 383.245.579,00, Rp 196.733.794,00, dan Rp 768.232.819,00.

5.3 Eksperimen

Pada sub bab ini akan dijelaskan mengenai hasil eksperimen yang dilakukan. Eksperimen dilakukan menggunakan metode eksak, menggunakan algoritma *hybrid cross entropy genetic algorithm*, serta menggunakan algoritma metaheuristik lainnya seperti *simulated annealing, cross entropy*, dan *modified genetic algorithm.*

5.3.1 Eksperimen dengan Metode Eksak

Pada subbab ini akan dijelaskan mengenai komputasi *single stage capcitated location problem* di PT. Petrokimia Gresik menggunakan metode eksak. Eksperimen ini dilakukan dengan bantuan *software* LINGO 11 untuk menyelesaikan model matematis yang telah divalidasi dan diverifikasi pada bab 4. Hasil eksperimen dengan metode eksak menggunakan bantuan *software* LINGO ditunjukkan pada Gambar 5.1 berikut.



Gambar 5. 1 *Solver Status* LINGO untuk *SSCWLP*

Berdasarkan Gambar 5.1 dapat diketahui bahwa komputasi menggunakan LINGO 11.0 menunjukkan hasil yang mencapai *global optimum* dengan metode *branch and bound.* Berdasarkan komputasi yang dilakukan didapatkan nilai optimum total biaya distribusi pupuk PT. Petrokimia Gresik setiap minggunya di Pulau Sumatera sebesar Rp 533.931.000,00. Berdasarkan hasil *running* LINGO ditunjukkan bahwa untuk mencapai kondisi optimum hanya dibutuhkan dua gudang lini 3 yang dibuka, yaitu GP Merangin dan GP Kotabumi, sedangkan pengiriman lain menuju lokasi *demand* dilayani langsung melalui gudang lini 2. Alokasi distribusi hasil komputasi menggunakan metode eksak secara rinci ditunjukkan pada Lampiran 4.

5.3.2 Eksperimen dengan Algoritma *Hybrid Cross Entropy – Genetic Algorithm*

Eksperimen menggunakan algoritma *hybrid* CE – GA dilakukan dengan cara melakukan beberapa kali *running* untuk beberapa kombinasi parameter yang digunakan dalam algoritma, antara lain parameter α, ρ, dan N. Dalam eksperimen yang dilakukan penulis, nilai maksimum iterasi yang ditetapkan sebesar 1000 iterasi, sedangkan jumlah replikasi untuk setiap kombinasi parameter dilakukan sebanyak 10 replikasi. Hal yang pertama dilakukan adalah melakukan komputasi dengan menguji beberapa nilai parameter α, yaitu 0.7, 0.8, dan 0.9. Hasil komputasi menggunakan beberapa nilai parameter α ditunjukkan pada Tabel 5.1 berikut.

Tabel 5. 1 Hasil Uji Parameter α

Alpha	*Rho*	N	Biaya	CPU time
0.7	0.2	15	Rp842,030,394.73	13.4137
			Rp852,421,436.24	10.4317
			Rp1,135,332,176.66	12.9014
			Rp615,259,710.00	10.8889
			Rp935,918,197.60	7.1916
			Rp1,229,654,901.95	10.4832
			Rp1,022,932,272.26	8.0911
			Rp645,000,501.12	10.1294
			Rp650,117,715.00	9.9125
			Rp703,702,796.20	9.0244
Rata-Rata			Rp863,237,010.18	10.2468
Standar Deviasi			Rp215,932,241.08	1.9201

Tabel 5. 1 Hasil Uji Parameter α (Lanjutan)

Alpha	*Rho*	*N*	*Biaya*	*CPU Time*
0.8	0.2	15	Rp613,227,120.00	12.9637
			Rp651,727,765.75	10.0777
			Rp1,071,409,091.65	13.9309
			Rp1,252,128,335.54	10.5769
			Rp1,059,606,668.81	11.0833
			Rp686,328,272.80	10.5882
			Rp931,210,033.66	12.4821
			Rp714,578,354.20	13.9221
			Rp753,038,611.32	10.1029
			Rp932,702,566.19	10.0057
Rata-Rata			Rp866,595,681.99	11.5734
Standar Deviasi			Rp214,798,872.34	1.5949
0.9	0.2	15	Rp719,876,584.40	7.1003
			Rp1,131,272,755.29	17.9193
			Rp969,612,776.00	10.9931
			Rp1,077,951,661.77	9.0028
			Rp1,056,256,322.14	8.7623
			Rp713,786,405.92	10.1107
			Rp642,147,300.00	9.0505
			Rp984,659,023.02	10.2294
			Rp952,023,390.41	8.2321
			Rp946,761,092.38	7.5882
Rata-Rata			Rp919,434,731.13	9.8989
Standar Deviasi			Rp168,755,877.52	3.0631

Berdasarkan hasil eksperimen pada Tabel 5.1 didapatkan bahwa nilai parameter α terbaik sebesar 0.7. Nilai ini selanjutnya digunakan sebagai nilai untuk melakukan pengujian parameter yang lain. Hal yang selanjutnya dilakukan adalah menguji nilai parameter ρ dengan beberapa nilai berbeda. Nilai yang digunakan dalam eksperimen ini adalah sebesar 0.2, 0.3 dan 0.4. Hasil uji parameter ρ yang dilakukan ditunjukkan dalam Tabel 5.2 Berikut.

Tabel 5. 2 Hasil Uji Parameter ρ

Alpha	*Rho*	N	Biaya	CPU *time*
0.7	0.2	15	Rp842,030,394.73	13.4137
			Rp852,421,436.24	10.4317
			Rp1,135,332,176.66	12.9014
			Rp615,259,710.00	10.8889
			Rp935,918,197.60	7.1916
			Rp1,229,654,901.95	10.4832
			Rp1,022,932,272.26	8.0911

Tabel 5. 2 Hasil Uji Parameter ρ (Lanjutan)

Alpha	*Rho*	N	Biaya	CPU *time*
			Rp645,000,501.12	10.1294
			Rp650,117,715.00	9.9125
			Rp703,702,796.20	9.0244
Rata-Rata			Rp863,237,010.18	10.2468
Standar Deviasi			Rp215,932,241.08	1.9201
0.7	0.3	15	Rp689,443,958.20	10.3897
			Rp685,191,480.00	8.7892
			Rp897,381,161.00	12.0745
			Rp628,274,494.80	11.4973
			Rp852,622,336.86	10.4209
			Rp1,001,779,470.35	8.0832
			Rp903,763,720.76	12.1002
			Rp744,569,899.12	10.9385
			Rp1,278,222,784.07	11.9736
			Rp690,013,573.80	17.0353
Rata-Rata			Rp837,126,287.90	11.3302
Standar Deviasi			Rp196,838,986.01	2.4267
0.7	0.4	15	Rp1,254,374,763.97	20.4261
			Rp1,016,348,885.70	9.4225
			Rp1,002,808,371.52	8.9233
			Rp728,839,382.00	10.1089
			Rp916,540,584.40	17.4409
			Rp942,381,965.98	14.4301
			Rp859,822,560.08	8.9545
			Rp1,165,610,762.89	12.1213
			Rp1,065,386,813.56	7.0200
			Rp693,534,390.00	10.8577
Rata-Rata			Rp964,564,848.01	11.9705
Standar Deviasi			Rp176,858,961.76	4.2371

Berdasarkan hasil eksperimen pada Tabel 5.2 didapatkan bahwa nilai parameter ρ terbaik sebesar 0.3. Nilai ini selanjutnya digunakan sebagai nilai untuk melakukan pengujian parameter yang lain. Hal yang selanjutnya dilakukan adalah menguji nilai parameter N dengan beberapa nilai berbeda. Nilai yang digunakan dalam eksperimen ini adalah sebesar 5, 10, 15, 20, dan 50. Hasil uji parameter N yang dilakukan ditunjukkan dalam Tabel 5.3 Berikut.

Tabel 5. 3 Hasil Uji Parameter N

Alpha	*Rho*	N	Biaya	CPU *time*
0.7	0.3	5	Rp1,292,219,624.92	3.0671
			Rp1,133,444,898.78	2.8934
			Rp1,117,511,299.72	1.9382

Tabel 5. 3 Hasil Uji Parameter N (Lanjutan)

Alpha	*Rho*	N	Biaya	CPU *time*
			Rp648,893,264.00	2.0571
			Rp1,031,486,351.45	2.5521
			Rp688,932,360.00	2.9551
			Rp944,089,698.40	2.0113
			Rp889,905,462.24	3.5881
			Rp846,021,661.95	4.0078
			Rp762,167,309.92	2.8856
Rata-Rata			Rp935,467,193.14	2.7956
Standar Deviasi			Rp208,677,001.16	0.6806
			Rp869,879,884.02	5.9280
			Rp713,449,298.00	6.9452
			Rp931,799,750.16	5.6332
			Rp1,020,248,731.74	7.9221
0.7	0.3	10	Rp753,829,174.80	5.0111
			Rp746,176,460.48	8.7729
			Rp1,028,051,942.72	6.8234
			Rp1,004,539,342.10	5.2113
			Rp610,431,350.00	9.7228
			Rp685,222,030.20	8.9889
Rata-Rata			Rp836,362,796.42	7.0959
Standar Deviasi			Rp153,845,295.86	1.6825
			Rp689,443,958.20	13.4137
			Rp685,191,480.00	10.4317
			Rp897,381,161.00	12.9014
			Rp628,274,494.80	10.8889
0.7	0.3	15	Rp852,622,336.86	7.1916
			Rp1,001,779,470.35	10.4832
			Rp903,763,720.76	8.0911
			Rp744,569,899.12	10.1294
			Rp1,278,222,784.07	9.9125
			Rp690,013,573.80	9.0244
Rata-Rata			Rp837,126,287.90	10.2468
Standar Deviasi			Rp196,838,986.01	1.9201
			Rp618,733,050.00	30.1862
			Rp729,588,950.60	9.6253
			Rp815,987,280.59	18.5485
			Rp636,083,220.00	27.2690
0.7	0.3	20	Rp1,182,483,659.50	11.1229
			Rp976,243,139.79	15.9589
			Rp718,110,433.20	13.7905
			Rp1,265,103,521.54	13.7437
			Rp1,158,603,669.89	12.6673
			Rp936,954,695.93	9.5941
Rata-Rata			Rp903,789,162.10	16.2506
Standar Deviasi			Rp236,866,954.46	7.1542

Tabel 5. 3 Hasil Uji Parameter N (Lanjutan)

Alpha	*Rho*	*N*	*Biaya*	*CPU time*
0.7	0.3	50	Rp 863,478,089.00	150.1859
			Rp 1,259,088,363.52	182.6928
			Rp 1,154,194,199.27	206.8105
			Rp 959,106,899.98	155.1586
			Rp 1,024,284,474.88	115.9243
			Rp 1,268,994,634.34	163.1146
			Rp 1,076,488,734.51	151.8514
			Rp 1,114,875,430.82	110.5579
			Rp 613,385,663.44	100.8858
			Rp 1,127,941,105.89	138.4197
Rata-Rata			Rp 1,046,183,759.57	147.56
Standar Deviasi			Rp 196,558,734.36	32.83

Berdasarkan hasil eksperimen yang dilakukan penulis pada Tabel 5.3 didapatkan bahwa nilai parameter N terbaik adalah dengan ukuran sampel sebanyak 10 sampel. Sehingga dapat disimpulkan bahwa kombinasi parameter terbaik didapatkan dengan nilai α sebesar 0.7, ρ sebesar 0.3, dan N sebanyak 10. Sedangkan nilai terbaik yang didapatkan dari komputasi menggunakan algoritma *hybrid* CE-GA menunjukkan total biaya distribusi sebesar Rp 610.431.350,00. Alokasi distribusi hasil komputasi menggunakan algoritma *hybrid* CE-GA secara rinci ditunjukkan pada Lampiran 5.

5.3.3 Eksperimen dengan Algoritma Metaheuristik Lain

Pada sub bab ini akan dilakukan eksperimen menggunakan algoritma metaheuristik yang lain sebagai pembanding dalam eksperimen. Algoritma metaheuristik yang digunakan dalam penelitian ini adalah algoritma *simulated annealing, genetic algorithm,* dan *cross entropy*. Berikut merupakan hasil eksperimen dengan algoritma metahuristik lain, yaitu:

- Eksperimen dengan Algoritma *Simulated Annealing*

  Eksperimen dengan algoritma SA dilakukan menggunakan algoritma yang dikembangkan oleh Kresna (2014). Komputasi ini dilakukan dengan nilai parameter optimal yang telah ditetapkan dalam Kresna (2014), yaitu dengan nilai temperatur awal sebesar 5000, faktor pereduksi temperatur sebesar 0.4, dan jumlah siklus sebesar 15. Hasil komputasi menggunakan algoritma *simulated annealing* ditunjukkan pada Tabel 5.4 berikut.

Tabel 5. 4 Hasil Komputasi dengan Algoritma SA

Replikasi	Biaya	CPU *Time*
1	Rp1,082,041,054.73	5.1324
2	Rp969,988,512.30	4.8048
3	Rp1,058,915,779.16	5.1324
4	Rp849,020,439.30	5.6784
5	Rp889,644,633.83	5.5692
6	Rp728,839,382.00	4.9920
7	Rp918,182,219.94	5.5068
8	Rp905,104,551.66	5.9280
9	Rp1,052,592,690.98	5.1324
10	Rp888,367,644.22	4.8360
Rata-Rata	Rp934,269,690.81	5.2713
Standar Deviasi	Rp109,202,178.56	0.3777

Berdasarkan Tabel 5.4 didapatkan bahwa dengan menggunakan algoritma *simulated annealing* solusi terbaik yang didapatkan menunjukkan total biaya distribusi sebesar Rp 728.839.382,00.

- Eksperimen dengan Algoritma *Genetic Algorithm*

  Eksperimen dengan algoritma *genetic algorithm* dilakukan menggunakan paremeter optimal yang didapatkan dari pengujian parameter algoritma *hybrid* CE – GA, yaitu menggunakan $\rho = 0.3$, N=10, dan maximum iterasi sebanyak 1000 iterasi. Hasil komputasi menggunakan algoritma *genetic algorithm* ditunjukkan pada Tabel 5.5 berikut.

Tabel 5. 5 Hasil Komputasi dengan Algoritma GA

Replikasi	Biaya	CPU *Time*
1	Rp1,235,773,163.60	30.1082
2	Rp889,055,203.72	39.0315
3	Rp1,168,641,936.81	47.4399
4	Rp1,010,043,000.05	37.6495
5	Rp1,110,137,214.04	33.9926
6	Rp1,138,159,281.00	34.1018
7	Rp925,864,575.20	34.1174
8	Rp1,287,361,709.20	33.9614
9	Rp998,406,520.88	34.0550
10	Rp959,730,082.30	33.4934
Rata-Rata	Rp1,072,317,268.68	35.7951
Standar Deviasi	Rp135,423,629.61	4.7431

Berdasarkan hasil komputasi pada Tabel 5.5 didapatkan bahwa dengan menggunakan algoritma *genetic algorithm* solusi terbaik yang didapatkan menunjukkan total biaya distribusi sebesar Rp 889.055.203,72.

- Eksperimen dengan Algoritma *Cross Entropy*

  Eksperimen dengan algoritma *cross entropy* dilakukan menggunakan paremeter optimal yang didapatkan dari pengujian parameter algoritma *hybrid* CE – GA, yaitu menggunakan α= 0.7, ρ = 0.3, N=10, dan maximum iterasi sebanyak 1000 iterasi. Hasil komputasi menggunakan algoritma *cross entropy* ditunjukkan pada Tabel 5.6 berikut.

Tabel 5. 6 Hasil Komputasi dengan Algoritma CE

Replikasi	Biaya	CPU *Time*
1	Rp1,914,482,455.25	56.4568
2	Rp2,036,831,354.08	56.6752
3	Rp1,824,008,135.25	56.5504
4	Rp1,865,090,886.77	56.5504
5	Rp1,953,204,354.32	56.8000
6	Rp1,965,183,624.29	56.9560
7	Rp2,023,695,348.77	56.8000
8	Rp1,914,696,247.27	57.0808
9	Rp1,947,537,321.85	57.0652
10	Rp1,990,569,108.92	56.7532
Rata-Rata	Rp1,943,529,883.68	56.7688
Standar Deviasi	Rp66,696,785.34	0.2170

Berdasarkan hasil komputasi pada Tabel 5.6 didpatkan bahwa dengan menggunakan algoritma *cross entropy* solusi terbaik yang didapatkan menunjukkan total biaya distribusi sebesar Rp 1.824.008.135,25. Solusi yang didapatkan dari algoritma CE dinilai tidak menunjukkan hasil yang baik karena hasil yang didapatkan tidak menunjukkan performansi yang lebih baik dibandingkan kondisi eksisting, melainkan lebih buruk dibandingkan kondisi eksisting yang hanya memerlukan biaya distribusi sebesar Rp 1.348.212.192,00.

5.4 Analisis Hasil Eksperimen

Pada sub bab ini akan dilakukan analisis hasil eksperimen yang telah dilakukan pada sub bab 5.3. Analisis dilakukan terhadap hasil eksperimen dengan metode eksak, algoritma *hybrid cross entropy – genetic algorithm,* dan algoritma metaheuristik lain yang meliputi algoritma *simulated annealing, genetic algorithm,* dan *cross entropy.*

5.4.1 Analisis Eksperimen dengan Metode Eksak

Komputasi menggunakan metode eksak mampu memberikan solusi untuk permasalahan *single stage capacitated warehouse location problem* dengan nilai yang mencapai *global optimum*. Dengan perhitungan *mixed integer linear programming* dengan metode *branch and bound* menggunakan *software* LINGO 11.0 didapatkan total biaya distribusi sebesar Rp 533.931.000,00. Hasil yang didapatkan menunjukkan penghematan sebesar Rp 814.281.192,00 atau sebesar 60,40%. Penghematan ini terjadi karena besarnya penghematan dari komponen biaya gudang dimana jumlah gudang penyangga lini 3 yang dibuka mengalami penurunan dari 29 gudang pada kondisi eksisting menjadi 2 gudang pada komputasi menggunakan metode eksak. Meskipun terdapat tambahan komponen biaya transportasi menuju lokasi *demand*, namun penurunan biaya gudang yang terjadi jauh lebih besar sehingga secara keseluruhan didapatkan penghematan biaya sebesar 60,40% tersebut.

Perbedaan hasil antara metode eksak dengan kondisi eksisting ini terjadi karena adanya perbedaan *objective* dalam pengambilan keputusan yang dilakukan. Pada kondisi eksisting, *objective* penentuan lokasi hanya didasarkan pada kedekatan dengan distributor sebagai konsumen akhir PT. Petrokimia Gresik tanpa memperhatikan faktor biaya secara rinci. Sedangkan pada komputasi menggunakan metode eksak hal yang diperhatikan adalah faktor finansial yaitu total biaya distribusi pupuk di Pulau Sumatera, sehingga hasil akhir dari kedua kondisi berbeda dan didapatkan penghematan yang besar dengan komputasi menggunakan metode eksak.

5.4.2 Analisis Eksperimen dengan Algoritma *Hybrid Cross Entropy – Genetic Algorithm*

Eksperimen dengan algoritma *hybrid cross entropy – genetic algorithm* dilakukan dengan cara melakukan pengujian untuk beberapa kombinasi parameter yang digunakan dalam algoritma CEGA. Parameter yang pertama diuji dalam eksperimen yang dilakukan penulis adalah parameter *smoothing* (α) yang menunjukkan besarnya pengaruh nilai iterasi sebelumnya terhadap iterasi saat ini. Pada eksperimen ini dilakukan pengujian terhadap tiga nilai α yaitu 0.7, 0.8, dan 0.9. Hasil komputasi yang didapat menunjukkan bahwa semakin besar nilai α

maka semakin jauh dari optimal fungsi tujuan yang didapatkan. Hal ini terjadi karena nilai α merupakan salah satu parameter yang mempengaruhi besarnya Pm atau probabilitas mutas. Semakin besar nilai α-nya maka semakin besar pula pengaruh Pm pada iterasi sebelumnya terhadap nilai Pm pada iterasi saat ini, sehingga nilai absolut perbedaan Pm iterasi saat ini dengan iterasi sebelumnya akan cenderung semakin kecil. Nilai absolut perbedaan Pm ini merupakan faktor penentu *stopping criteria* dalam algoritma *hybrid* CE – GA, sehingga semakin besar nilai α maka akan semakin cepat *stopping criteria* tercapai sehingga waktu komputasi akan semakin cepat, namun hasil komputasi yang dihasilkan semakin menjauhi optimal. Sehingga berdasarkan percobaan yang dilakukan didapatkan parameter α yang paling tepat sebesar 0.7.

Tahapan selanjutnya adalah melakukan pengujian terhadap parameter ρ atau proporsi sampel elite. Pada eksperimen yang dilakukan penulis digunakan tiga nilai ρ sebagai dasar pengujian yaitu 0.2, 0.3, dan 0.4. Secara umum, semakin besar nilai ρ maka semakin besar penjaminan nilai yang mendekati optimal berada pada iterasi selanjutnya. Namun, apabila nilai ρ yang digunakan terlalu besar, maka kemungkinan komputasi terjebak dalam lokal optimum akan semakin besar, sehingga solusi yang lebih dekat terhadap nilai global optimum tidak didapatkan. Oleh karena itu, dalam eksperimen yang dilakukan pada sub bab 5.3 ditunjukkan bahwa untuk menyelesaikan kasus *single stage capacitated warehouse location problem* nilai parameter ρ yang paling tepat sebesar 0.3.

Parameter ketiga yang diuji dalam eksperimen ini adalah parameter N atau ukuran sampel, dimana dilakukan pengujian untuk empat ukuran sampel yaitu 5, 10, 15, dan 20. Pada hakikatnya, semakin besar ukuran sampel, maka semakin baik pula hasil komputasi yang dihasilkan, namun waktu komputasi yang dibutuhkan juga akan meningkat karena meningkatnya nilai yang harus dievaluasi dalam setiap langkah algoritma. Semakin kecil ukuran sampel yang dibangkitkan maka nilai random yang dibangkitkan juga semakin sedikit sehingga kemungkinan solusi yang mendekati optimal ikut dibangkitkan juga akan semakin kecil. Akan tetapi, pada eksperimen yang dilakukan penulis didapatkan solusi algoritma yang lebih baik pada ukuran sampel 10. Hal ini dapat terjadi karena *stopping criteria* dalam algoritma *hybrid* CE – GA ditentukan oleh Pm yang

ditentukan oleh sampel elite. Semakin banyak sampel elite dalam setiap iterasi secara tidak langsung akan mempengaruhi nilai *u* pada persamaan 3.3 yang akan berpengaruh terhadap nilai Pm sehingga terdapat kemungkinan iterasi dianggap konvergen meskipun sebelumnya belum terjadi konvergensi. Namun, karena nilai yang harus dievaluasi untuk tiap langkah semakin banyak, meskipun konvergensi lebih cepat dicapai dalam segi jumlah langkah, namun waktu komputasinya tetap semakin lama.

5.4.3 Analisis Perbandingan Hasil Algoritma *Hybrid* CE – GA dengan Kondisi Eksisting

Berdasarkan eksperimen yang dilakukan pada sub bab 5.3, solusi terbaik untuk hasil komputasi menggunakan metode *hybrid* CE - GA menunjukkan total biaya distribusi sebesar Rp 610.431.350,00. Nilai ini menunjukkan penghematan biaya sebesar Rp 737.780.842,00 atau penghematan sebesar 54,73%. Pada komputasi menggunakan algoritma *hybrid* CE – GA terdapat penurunan jumlah gudang yang dibuka dibandingkan kondisi eksisting menjadi empat gudang lini 3. Gudang lini 3 yang dibuka antara lain adalah GP Solok, GP Bukittinggi, GP Merangin, dan GP Kotabumi. Alokasi distribusi pupuk hasil komputasi menggunakan algoritma *hybrid* CE – GA ditunjukkan pada Lampiran 5.

Reduksi biaya ini terjadi karena penurunan komponen biaya gudang yang cukup tinggi diakibatkan adanya penurunan jumlah gudang lini 3 yang dibuka dari 29 gudang menjadi 4 gudang. Selain itu, seperti halnya metode eksak, erbedaan hasil antara komputasi menggunakan algoritma *hybrid* CE – GA dengan kondisi eksisting ini terjadi karena adanya perbedaan *objective* dalam pengambilan keputusan yang dilakukan. Pada kondisi eksisting, *objective* penentuan lokasi hanya didasarkan pada kedekatan dengan distributor sebagai konsumen akhir PT. Petrokimia Gresik tanpa memperhatikan faktor biaya secara rinci. Sedangkan pada komputasi menggunakan metode eksak dan algortima *hybrid* CE – GA hal yang diperhatikan adalah faktor finansial yaitu total biaya distribusi pupuk di Pulau Sumatera, sehingga hasil akhir dari kedua kondisi berbeda dan didapatkan penghematan yang besar dengan komputasi menggunakan algoritma *hybrid* CE - GA.

5.4.4 Analisis Perbandingan Hasil Algoritma *Hybrid* CE – GA dengan Metode Eksak

Berdasarkan eksperimen yang dilakukan pada sub bab 5.3, komputasi menggunakan metode eksak menunjukkan biaya yang lebih optimum dibandingkan komputasi menggunakan algoritma *hybrid* CE – GA. Menggunakan metode eksak didapatkan total biaya distribusi sebesar Rp 533.931.000,00. Hasil yang didapatkan dari komputasi metode eksak menunjukkan penghematan sebesar Rp 814.281.192,00 atau sebesar 60,40%. Sedangkan komputasi menggunakan algoritma *hybrid* CE – GA menunjukkan total biaya distribusi sebesar Rp 610.431.350,00 dengan penghematan biaya dibandingkan metode eksak sebesar Rp 737.780.842,00 atau penghematan sebesar 54,73%. Selisih biaya yang dihasilkan oleh kedua metode tersebut bernilai sebesar Rp 76.500.350,00. Selisih biaya ini menunjukkan adanya *potential losses*, namun *potential losses* ini dirasa cukup kecil dibandingkan kondisi eksisting permasalahan yang mencapai Rp 1.348.212.192,00 per minggu.

5.4.5 Analisis Perbandingan Hasil Algoritma *Hybrid* CE – GA dengan Algoritma Metaheuristik Lain

Pada sub bab ini akan dilakukan analisis perbandingan antara hasil komputasi algoritma *hybrid* CE – GA dengan hasil komputasi algoritma SA, CE, dan GA. Ringkasan hasil komputasi keempat algoritma tersebut ditunjukkan dalam Tabel 5.7 berikut.

Tabel 5. 7 Perbandingan Hasil Komputasi Beberapa Algoritma Metaheuristik

Algoritma	CE - GA	SA	GA	CE
Rata-Rata Nilai Solusi	Rp836,362,796	Rp934,269,691	Rp1,072,317,269	Rp1,943,529,884
Standar Deviasi	Rp153,845,296	Rp109,202,179	Rp135,423,630	Rp66,696,785
Solusi Terbaik	Rp610,431,350	Rp728,839,382	Rp889,055,204	Rp1,824,008,135
Gap dengan solusi optimal	Rp76,500,350	Rp194,908,382	Rp355,124,204	Rp1,290,077,135
% Gap dengan nilai optimal	9.39%	23.94%	43.61%	158.43%
CPU Time	7.0959	5.2713	35.7951	56.7688

Tabel 5.7 menunjukkan perbandingan performansi algoritma *hybrid* CE – GA, SA, GA, dan CE dalam menyelesaikan *single stage capacitated warehouse location problem.* Keempat algoritma ini menunjukkkan hasil yang berbeda secara signifikan dalam menyelesaikan SSCWLP. Solusi terbaik ditunjukkan oleh algoritma *hybrid* CE – GA dengan total biaya sebesar Rp 610.431.350,00. Solusi yang ditunjukkan oleh algoritma *hybrid* CE – GA menunjukkan hasil yang lebih baik dibandingkan komputasi menggunakan algoritma *simulated annealing* yang dikembangkan oleh Kresna (2014) baik dilihat dari solusi terbaiknya maupun rata-rata nilai solusi yang ditunjukkan. Akan tetapi, dari segi waktu komputasi algoritma SA menunjukkan waktu komputasi yang lebih cepat dibandingkan algoritma *hybrid* CE – GA. Perbedaan ini dapat terjadi karena karakteristik algoritma *hybrid* CE – GA yang bersifat *population based,* dimana dibangkitkan beberapa solusi untuk setiap langkah, berbeda dengan algoritma SA yang hanya membangkitkan satu solusi untuk setiap langkah. Karena dalam algoritma SA hanya dibangkitkan satu solusi untuk setiap langkah, maka waktu komputasi juga semakin cepat karena nilai yang harus dievaluasi dalam setiap langkah lebih sedikit. Namun, di sisi lain, pembangkitan satu solusi setiap langkah ini mengakibatkan kemungkinan solusi optimal ikut dibangkitkan lebih kecil dibandingkan algoritma *population based* dimana dalam satu langkah terdapat beberapa solusi yang dibangkitkan sehingga nilai solusi yang mungkin dihasilkan lebih tersebar dibandingkan algoritma yang *non-population based* seperti *simulated annealing.*

Penggabungan algoritma *cross entropy* dengan *genetic algorithm* juga menunjukkan hasil komputasi yang lebih baik dibandingkan komputasi menggunakan algoritma *cross entropy* asli maupun *genetic algorithm* tanpa *crossover.* Penggabungan kedua algoritma ini menunjukkan performansi yang lebih baik, baik dari segi waktu komputasi maupun solusi yang dihasilkan. Hal ini dapat terjadi karena dalam penggabungan kedua algoritma, langkah-langkah yang menyebabkan komputasi berlangsung lama dan menyebabkan solusi yang dibangkitkan menjauh dari nilai optimal seperti mekanisme pembangkitan solusi pada CE dan *crossover* pada GA dihilangkan. Sehingga hasil komputasi yang

dihasilkan dari algoritma *hybrid* CE – GA menujukkan performansi yang lebih baik dibandingkan algoritma CE dan GA.

BAB 6
KESIMPULAN

Pada bab ini akan dilakukan penarikan kesimpulan terkait hasil eksperimen yang telah dilakukan. Setelah itu akan diberikan saran-saran yang dapat dijadikan sebagai rekomendasi untuk acuan penelitian selanjutnya.

6.1 Kesimpulan

Berikut merupakan beberapa kesimpulan yang dapat diambil dalam penelitian ini, antara lain:

1. Dalam penelitian ini dapat dihasilkan model algoritma *hybrid cross entropy - genetic algorithm* dalam menyelesaikan *single stage capacitated warehouse location problem*
2. Berdasarkan hasil komputasi menggunakan algoritma *hybrid cross entropy – genetic algorithm* didapatkan alternatif solusi dengan total biaya distribusi sebesar Rp 610.431.350,00. Hasil alternatif solusi yang dihasilkan menunjukkan penghematan sebesar Rp 737.780.842,00 atau penghematan sebesar 54,73% dari konsidi eksisting dengan membuka empat gudang lini 3, antara lain GP Solok, GP Bukittinggi, GP Merangin, dan GP Kotabumi.
3. Algoritma *hybrid cross entropy – genetic algorithm* mampu menunjukkan alternatif solusi yang lebih baik dibandingkan algortima *simulated annealing, cross entropy,* dan *genetic algorithm* dalam menyelesaikan *single stage capacitated warehouse location problem*.

6.2 Penelitian Selanjutnya

Berikut merupakan beberapa alternatif yang dapat dijadikan acuan dalam penelitian selanjutnya, antara lain:

1. Menyelesaikan permasalahan *single stage capacitated warehouse locaton problem* dengan beberapa ukuran set data dan ukuran set data yang lebih besar untuk mengetahui performansi algoritma dalam ukuran kasus berbeda
2. Memperluas ruang lingkup penelitian *single stage capacitated warehouse location problem* pada PT. Petrokimia Gresik, antara lain cakupan wilayah

menjadi seluruh Indonesia dan mempertimbangkan faktor pengiriman dari gudang lini 1, sehingga didapatkan alokasi distribusi secara keseluruhan.

DAFTAR PUSTAKA

Aardal, K., Berd, P. L., Gijswijt, D., and Li, S., 2015. 'Approximation algorithms for hard capacitated k-facility location problems'. European Journal of Operational Research, Vol. 242 pp. 358 – 368.

Alizadeh, M., Mahdavi, I., Mahdavi-Amiri, I., and Shiripour, S., 2015. 'A Capacitated Location-Allocation Problem with Stochastic Demands Using Sub-Sources: An Empirical Study'. Applied Soft Computing, Vol. 34 pp. 551-571.

Ballou, R. H., 2003. *Business Logistic / Supply Chain Management.* New Jersey: Prentice Hall.

Berman, O., and Krass, D., 2002, 'Facility Location Problems with Stochastics Demands and Congestion', in Z Drezner and H W Hamacher (eds), *Facility Location: Applications and Theory,* Springer-Verlag, Germany, pp. 329-371.

Chopra, S., and Meindl, P., 2001. *Supply Chain Management: Strategy, Planning, and Operation.* New Jersey: Prentice Hall.

Daskin, M.S., 2013. *Network and Discrete Location: Models, Algorithms, and Applications.* New York: John Wiley and Sons.

Eiselt, H.A., and Marianov, V., 2011. *Foundations of Location Analysis.* New York: Springer Science and Business Media.

Guastaroba, G., and Speranza, M. G., 2014. 'A Heuristic for BILP Problems: The Single Source Capacitated Facility Location Problem'. European Journal of Operational Research Vol. 238 pp. 438-450.

Khumalawa, B.M., 1972. 'An Efficient Branch and Bound Algorithm for the Warehouse Locarion Problem'. Management Science, Vol. 18 No. 12.

Kresna, I.G.N.A., 2014 'Algoritma Simulated Annealing untuk Menyelesaikan Single Stage Capacitated Warehouse Location Problem'. Undergraduate Thesis. Institut Teknologi Sepuluh Nopember.

Peng, R., Rui-hua, X., Jin, Q., 2008. 'Bi-level Simulated Annealing Algorithm for Facility Location Problem', International Conference on Information Management, Innovation Management and Industrial Engineering

Pujawan, I. N., 2005. *Supply Chain Management.* Surabaya: Guna Widya.

Rahmaniani, R., and Ghaderi, A., 2015. 'An Algorithm with Different Exploration Mechanism: Experimental Results to Capacitated Facility Location/Network Design Problems'. Expert Systems with Apllication, Vol. 42 pp. 3790-3800.

Razali, N.M., 2015. 'An Efficient Genetic Algorithm for Large Scale Vehicle Routing Problem Subject to Precedence Constraints'. Procedia – Social and Behavioral Sciences, Vol. 195 pp. 1922 -1931.

Santosa, B., and Damayanti, R., 2014. 'The Development of Hybrid Cross Entropy – Genetic Algorithm for Multi Product Inventory Ship Routing Problem with Heterogeneous Fleet', Proceedings of the 2014 International Conference on Industrial Engineering and Operations Management, Bali, Indonesia, January 7 – 9.

Santosa, B., and Hardiansyah, N., 2010. Cross Entropy Method for Solving Generelixed Orienteering Problem. iBusiness, Vol. 2 pp. 342-347.

Santosa, B., and Nurkhalida, L., 2012. 'A Cross Entropy – Genetic Algorithm Approach for Multi Objective Job Shop Scheduling Problem', Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management, Istanbul, Turkey, July 3 – 6.

Santosa, B., and Willy, P., 2011. *Metoda Metaheuristik: Konsep dan Implementasi.* Surabaya: Guna Widya.

Sharma, R.R.K., and Berry, V., 2007. 'Developing New Formulations and Relaxations of Single Stage Capacitated Warehouse Location Problem (SSCWLP): Empirical Investigation for Assessing Relative Strengths and Computational Effort'. European Journal of Operational Research, Vol. 177 pp. 803 – 812.

Verma, P., and Sharma, R.R.K., 2011. 'Vertical Decomposition Approach to Solve Single Stage Capacitated Warehouse Location Problem'. American Journal of Operational Research, Vol. 1 pp. 100-117.

Yang, Z., Chu, F., and Chen, H., 2012. 'A Cut and Solve Based Algorithm for the Single Source Capacitated Facility Location Problem'. European Journal of Operational Research, Vol. 221 pp. 521-532.

LAMPIRAN 1
DATA EKSISTING

Lokasi dan Kapasitas Gudang Lini 2

Gudang Lini 2	Kapasitas (ton)
GP Lhokseumawe	5,000
GP Medan	88,000
DC Padang	45,000
GP Dumai	10,000
GP Tanjung Pinang	600
GP Jambi	8,000
GP Palembang	26,000
GP Bengkulu	7,000
DC Lampung	70,600
GP Pangkal Pinang	6,100

Lokasi dan Kapasitas Gudang Lini 3

Gudang Lini 3	Kapasitas (ton)
GP Kotacane	800
GP Blang Pidie	1,000
GP Aceh Tamiang	1,275
GP Sibolga Tapanuli	1,500
GP Balige	1,000
GP Asahan	2,500
GP Simalungun	1,500
GP Dairi	1,300
GP Tanah Karo	2,500
GP Serdang Bedagai	2,000
GP Labuhan Batu	1,200
GP Padang Sidempuan	1,000
GP Painan	1,500
GP Solok	2,000
GP Batu Sangkar	1,000
GP Pasaman Barat	1,000
GP Bukit Tinggi	1,000
GP Payakumbuh	1,500
GP Indra Giri Hulu	2,500
GP Pekan Baru	2,600
GP Kerinci	1,000
GP Merangin	2,000
GP Lahat	2,000

Gudang Lini 3	Kapasitas (ton)
GP Martapura	6,000
GP Lubuk Linggau	1,000
GP Pringsewu	2,500
GP Bandar Jaya	3,500
GP Gunung Sugih	10,000
GP Kotabumi	2,000

Komponen Biaya Gudang pada Gudang Lini 3

Gudang Lini 3	Biaya Sewa Gudang dan Biaya Tenaga Kerja	Biaya Bongkat Muat (/ton)
GP Kotacane	IDR4,593,750	IDR28,090
GP Blang Pidie	IDR7,200,000	IDR30,210
GP Aceh Tamiang	IDR5,000,000	IDR30,000
GP Sibolga Tapanuli	IDR5,421,875	IDR12,075
GP Balige	IDR4,014,044	IDR15,488
GP Asahan	IDR7,412,500	IDR15,645
GP Simalungun	IDR4,931,875	IDR15,698
GP Dairi	IDR5,411,750	IDR15,635
GP Tanah Karo	IDR5,076,250	IDR14,385
GP Serdang Bedagai	IDR4,910,000	IDR14,385
GP Labuhan Batu	IDR4,262,500	IDR15,200
GP Padang Sidempuan	IDR4,379,313	IDR15,745
GP Painan	IDR6,423,338	IDR14,295
GP Solok	IDR6,183,950	IDR8,775
GP Batu Sangkar	IDR4,517,600	IDR14,400
GP Pasaman Barat	IDR4,738,125	IDR15,613
GP Bukit Tinggi	IDR5,380,125	IDR10,285
GP Payakumbuh	IDR5,738,500	IDR12,720
GP Indra Giri Hulu	IDR6,108,350	IDR31,900
GP Pekan Baru	IDR7,575,000	IDR31,900
GP Kerinci	IDR4,562,500	IDR16,000
GP Merangin	IDR5,687,500	IDR11,200
GP Lahat	IDR7,743,750	IDR10,945
GP Martapura	IDR14,043,750	IDR11,400
GP Lubuk Linggau	IDR4,810,250	IDR11,405
GP Pringsewu	IDR7,625,000	IDR10,000
GP Bandar Jaya	IDR9,348,450	IDR9,075
GP Gunung Sugih	IDR26,500,000	IDR8,500
GP Kotabumi	IDR7,133,750	IDR9,240

Lokasi dan Jumlah *Demand*

Lokasi	Demand
Toba Samosir	38
Kota Dumai	47
Ogan Komering Ulu Selatan	458
Kab. Kepahiang	66
Kab. Dharmasraya	465
Kab Pesawaran	724
Ogan Komering Ulu Timur	336
Kab Ogan Ilir	344
Kab Empat Lawang	185
Gunung Sitoli	29
Aceh Singkil Utara	245
Bintan	20
Kota Tanjung Pinang	12
Kota Batam	7
Karimun Tanung	8
Kota Pangkal Pinang	13
Belitung Timur	150
Kab. Bangka Barat	245
Bangka Tengah	284
Bangka Selatan	196
Belitung	110
Bangka	270
Pesisir Barat	428
Kota Metro	250
Kota Bandar Lampung	88
Tulang Bawang Barat	240
Mesuji	520
Pringsewu	340
Tulang Bawang	994
Way Kanan	700
Lampung Utara	436
Lampung Tengah	1276
Lampung Timur	1576
Lampung Selatan	1142
Kab. Tanggamus	905
Lampung Barat	477
Kota Bengkulu	48
Bengkulu Tengah	235

Lokasi	Demand
Lebong	242
Mukomuko	1133
Seluma	272
Kaur	370
Kab. Bengkulu Utara	849
Rejang Lebong	399
Bengkulu Selatan	341
Musi Rawas Utara	192
Kab. Panukal. Abab Lem. Ilir	76
Kota Lubuk Linggau	64
Kota Pagar Alam	126
Kota Prabumulih	22
Kota Palembang	24
Banyuasin	460
Musi Banyuasin	878
Musi Rawas	404
Lahat	146
Muara Enim	231
Ogan Komering Ilir	645
Ogan Komering Ulu	232
Sungai Penuh	130
Kota Jambi	16
Bungo	344
Tebo	481
Tanjung Jabung Barat	384
Tanjung Jabung Timur	536
Muaro Jambi	256
Kab. Batang Hari	256
Sarolangun	314
Merangin	916
Kerinci	557
Kep. Meranti	110
Kota Pekan Baru	32
Rokan Hilir	216
Bengkalis	337
Rokan Hulu	269
Kampar	390
Pelalawan	430
Kab. Indragiri Hilir	258
Kab. Indragiri Hulu	555

Lokasi	Demand
Kuantan Singingi	347
Kota Pariaman	100
Kota Payakumbuh	100
Kota Bukittinggi	15
Kota Padang Panjang	15
Kota Sawah Lunto	28
Kota Solok	30
Kota Padang	332
Pasaman barat	855
Kab. Solok Selatan	1033
Pasaman	264
Lima Puluh Kota	333
Agam	580
Padang Pariaman	300
Tanah Datar	428
Sijunjung	120
Solok	30
Pesisir Selatan	454
Labuan Batu Utara	129
Labuan Batu Selatan	250
Serdang bedagai	580
Kota Padang Sidempuan	75
Kota Binjai	40
Kota Medan	19
Kota Tebing Tinggi	25
Kota Pematang Siantar	175
Padang Lawas Utara	931
Kab. Batu Bara	200
Samosir	125
Pakpak Bharat	100
Humbang Hasundutan	100
Langkat	125
Deli Serdang	250
Karo	240
Dairi	250
Simalungun	460
Kab. Asahan	225
Labuhan Batu	104
Tapanuli Utara	191
Tapanuli Tengah	118

Lokasi	Demand
Tapanuli Selatan	150
Mandailing Natal	100
Nias	20
Kota Subulussalam	397
Kota Langsa	45
Aceh Tamiang	65
Aceh Timur	50
Aceh Singkil	245

Biaya Pengiriman dari Gudang Lini 2 Menuju Gudang Lini 3

	GP Kotacane	GP Blang Pidie	GP Aceh Tamiang	GP Sibolga Tapanuli	GP Balige	GP Asahan	GP Simalungun
GP Lhokseumawe	IDR13,342.40	IDR10,472.80	IDR4,111.60	IDR15,952.40	IDR11,010.40	IDR9,864.00	IDR9,508.00
GP Medan	IDR7,669.20	IDR14,381.60	IDR4,111.60	IDR10,279.20	IDR5,337.20	IDR4,190.80	IDR3,834.80
DC Padang	IDR17,558.00	IDR24,270.40	IDR14,000.40	IDR20,168.00	IDR15,226.00	IDR14,079.60	IDR13,723.60
GP Dumai	IDR18,479.20	IDR25,191.60	IDR14,921.60	IDR21,089.20	IDR16,147.20	IDR15,000.80	IDR14,644.80
GP Tanjung Pinang	IDR38,232.80	IDR44,945.20	IDR34,675.20	IDR40,842.80	IDR35,900.80	IDR34,754.40	IDR34,398.40
GP Jambi	IDR27,952.00	IDR34,664.40	IDR24,394.40	IDR30,562.00	IDR25,620.00	IDR24,473.60	IDR24,117.60
GP Palembang	IDR37,836.00	IDR44,548.40	IDR34,278.40	IDR40,446.00	IDR35,504.00	IDR34,357.60	IDR34,001.60
GP Bengkulu	IDR33,372.40	IDR40,084.80	IDR29,814.80	IDR35,982.40	IDR31,040.40	IDR29,894.00	IDR29,538.00
DC Lampung	IDR25,465.20	IDR32,177.60	IDR21,907.60	IDR28,075.20	IDR23,133.20	IDR21,986.80	IDR21,630.80
GP Pangkal Pinang	IDR58,232.80	IDR60,945.20	IDR50,675.20	IDR56,842.80	IDR51,900.80	IDR50,754.40	IDR50,398.40

	GP Dairi	GP Tanah Karo	GP Serdang Bedagai	GP Labuhan Batu	GP Padang Sidem	GP Painan	GP Solok
GP Lhokseumawe	IDR12,176.80	IDR9,970.40	IDR8,835.60	IDR11,840.80	IDR15,562.00	IDR26,180.40	IDR25,728.80
GP Medan	IDR6,503.60	IDR4,297.20	IDR3,162.40	IDR6,167.60	IDR9,888.80	IDR20,507.20	IDR20,055.60
DC Padang	IDR16,392.40	IDR14,186.00	IDR13,051.20	IDR16,056.40	IDR9,888.80	IDR3,700.40	IDR3,248.80
GP Dumai	IDR17,313.60	IDR15,107.20	IDR13,972.40	IDR16,977.60	IDR12,166.00	IDR15,866.40	IDR15,414.80
GP Tanjung Pinang	IDR37,067.20	IDR34,860.80	IDR41,769.84	IDR44,294.21	IDR40,252.46	IDR43,360.80	IDR42,981.46
GP Jambi	IDR26,786.40	IDR24,580.00	IDR23,445.20	IDR26,450.40	IDR20,282.80	IDR14,094.40	IDR13,642.80
GP Palembang	IDR36,670.40	IDR34,464.00	IDR33,329.20	IDR36,334.40	IDR30,166.80	IDR23,978.40	IDR23,526.80
GP Bengkulu	IDR32,206.80	IDR30,000.40	IDR28,865.60	IDR31,870.80	IDR25,703.20	IDR24,457.26	IDR24,005.66
DC Lampung	IDR24,299.60	IDR22,093.20	IDR20,958.40	IDR23,963.60	IDR17,796.00	IDR37,201.20	IDR36,749.60
GP Pangkal Pinang	IDR53,067.20	IDR50,860.80	IDR49,726.00	IDR52,731.20	IDR47,919.60	IDR51,620.00	IDR51,168.40

	GP Batu Sangkar	GP Pasaman Barat	GP Bukit Tinggi	GP Payakumbuh	GP Indra Giri Hulu	GP Pekan Baru	GP Kerinci
GP Lhokseumawe	IDR26,265.20	IDR27,074.80	IDR25,958.80	IDR26,396.00	IDR45,787.20	IDR30,585.60	IDR47,500.00
GP Medan	IDR20,592.00	IDR21,401.60	IDR20,285.60	IDR20,722.80	IDR33,195.72	IDR24,912.40	IDR35,981.11
DC Padang	IDR3,785.20	IDR4,594.80	IDR3,478.80	IDR3,916.00	IDR16,712.33	IDR8,105.60	IDR8,019.20
GP Dumai	IDR15,951.20	IDR16,760.80	IDR15,644.80	IDR16,082.00	IDR8,380.40	IDR6,708.40	IDR20,185.20
GP Tanjung Pinang	IDR43,432.03	IDR44,112.10	IDR43,174.66	IDR43,541.90	IDR37,072.56	IDR35,668.08	IDR46,988.59
GP Jambi	IDR14,179.20	IDR14,988.80	IDR13,872.80	IDR14,310.00	IDR7,452.80	IDR18,499.60	IDR10,807.60
GP Palembang	IDR24,063.20	IDR24,872.80	IDR23,756.80	IDR24,194.00	IDR23,344.16	IDR28,383.60	IDR22,338.32
GP Bengkulu	IDR24,542.06	IDR25,351.66	IDR24,235.66	IDR24,672.86	IDR23,761.28	IDR28,693.31	IDR14,155.88
DC Lampung	IDR37,737.60	IDR38,010.80	IDR36,085.20	IDR37,638.40	IDR34,940.48	IDR37,316.57	IDR31,135.30
GP Pangkal Pinang	IDR51,704.80	IDR52,514.40	IDR51,398.40	IDR51,835.60	IDR44,134.00	IDR42,462.00	IDR55,938.80

	GP Merangin	GP Lahat	GP Martapura	GP Lubuk Linggau	GP Pringsewu	GP Bandar Jaya	GP Gunung Sunggih	GP Kotabumi
GP Lhokseumawe	IDR35,718.44	IDR61,483.24	IDR68,071.68	IDR55,363.28	IDR74,511.60	IDR63,088.60	IDR75,248.88	IDR73,724.08
GP Medan	IDR23,042.15	IDR51,004.56	IDR56,203.92	IDR44,297.88	IDR62,272.80	IDR51,157.04	IDR62,898.00	IDR61,182.60
DC Padang	IDR9,214.00	IDR26,483.04	IDR33,050.04	IDR21,652.16	IDR41,003.12	IDR26,569.64	IDR39,111.12	IDR37,353.94
GP Dumai	IDR21,380.00	IDR35,108.52	IDR41,742.66	IDR30,371.54	IDR47,712.04	IDR35,294.30	IDR46,163.32	IDR44,447.92
GP Tanjung Pinang	IDR47,992.22	IDR59,524.18	IDR65,096.86	IDR55,545.12	IDR70,111.14	IDR59,680.24	IDR68,810.21	IDR67,369.28
GP Jambi	IDR5,890.80	IDR15,531.12	IDR22,626.51	IDR11,217.63	IDR24,587.40	IDR16,162.88	IDR28,666.24	IDR26,874.60
GP Palembang	IDR30,028.77	IDR5,673.20	IDR5,080.40	IDR8,065.20	IDR12,971.68	IDR8,355.29	IDR12,058.56	IDR10,368.64
GP Bengkulu	IDR29,085.56	IDR8,279.85	IDR11,385.60	IDR6,128.00	IDR19,228.61	IDR8,393.44	IDR19,497.31	IDR18,183.24
DC Lampung	IDR39,260.64	IDR9,856.00	IDR5,515.20	IDR14,430.40	IDR2,239.60	IDR2,277.20	IDR2,277.20	IDR2,846.40
GP Pangkal Pinang	IDR57,133.60	IDR70,862.12	IDR77,496.26	IDR66,125.14	IDR83,465.64	IDR71,047.90	IDR81,916.92	IDR80,201.52

LAMPIRAN 2
LINGO SCRIPT

```
sets:
gudang/1..10/:supply;
dc/1..29/:kapasitas,sewa,ulcost,opendc,masuk;
cust/1..126/:demand;
kirim_g_dc(gudang,dc):biaya_g_dc,volume_g_dc;
kirim_dc_c(dc,cust):jarak_dc_c,volume_dc_c,bin_dc_c;
kirim_g_c(gudang,cust):jarak_g_c,volume_g_c,bin_g_c;
endsets

data:
supply, demand, kapasitas, sewa, ulcost, biaya_g_dc,
jarak_dc_c, jarak_g_c=@ole('C:\Users\KOI
2\Desktop\Fatmah\input lingo.xlsx');
@ole('C:\Users\KOI 2\Desktop\Fatmah\output
lingo.xlsx')=volume_g_dc, volume_dc_c, volume_g_c;
b=89.33;

enddata

!constraint supply;
@for(gudang(i):@sum(dc(j):volume_g_dc(i,j))+@sum(cust(k):vol
ume_g_c(i,k))<=supply(i));
!constraint demand;
@for(cust(k):@sum(dc(j):volume_dc_c(j,k))+@sum(gudang(i):vol
ume_g_c(i,k))>=demand(k));
!constraint kapasitas (inbound);
@for(dc(j):@sum(gudang(i):volume_g_dc(i,j))<=kapasitas(j)*op
endc(j));
!constraint kapasitas (outbound);
@for(dc(j):@sum(cust(k):volume_dc_c(j,k))<=kapasitas(j)*open
dc(j));
!keseimbangan dc;
@for(dc(j):@sum(cust(k):volume_dc_c(j,k))=@sum(gudang(i):vol
ume_g_dc(i,j)));
!binary constraint;
@for(dc(j):@bin(opendc(j)));
!jumlah masuk dc;
@for(dc(j):@sum(gudang(i):volume_g_dc(i,j))=masuk(j));
!setiap lokasi demand dipasok dari satu gudang;
@for(kirim_dc_c(j,k):volume_dc_c(j,k)<=99999999*bin_dc_c(j,k
));
@for(kirim_g_c(i,k):volume_g_c(i,k)<=99999999*bin_g_c(i,k));
@for(cust(k):@sum(kirim_dc_c(j,k):bin_dc_c(j,k))+@sum(kirim_
g_c(i,k):bin_g_c(i,k))=1);
@for(kirim_dc_c(j,k):@bin(bin_dc_c(j,k)));
@for(kirim_g_c(i,k):@bin(bin_g_c(i,k)));
```

```
!fungsitujuan [biaya transportasi 1-2 + biaya transportasi
1-3 + biaya transportasi 2-3 + biaya sewa gudang];
min=@sum(kirim_g_dc(i,j):biaya_g_dc(i,j)*volume_g_dc(i,j))+@
sum(kirim_dc_c(j,k):jarak_dc_c(j,k)*volume_dc_c(j,k)*b)+@sum
(kirim_g_c(i,k):jarak_g_c(i,k)*volume_g_c(i,k)*b)+@sum(dc(j)
:masuk(j)*ulcost(j))+@sum(dc(j):opendc(j)*sewa(j));
```

LAMPIRAN 3
KODE PROGRAM MATLAB

Kode Program Algortima *Hybrid Cross Entropy – Genetic Algorithm* untuk Menyelesaikan Single Stage Capacitated Warehouse Location Problem

```
function [mincost, alokasi_min]=CEGA_SSCWLP

%input%
[N, kapasitas, demand, suplai, jarak23, jarak13, biaya12,
biayasewa, ULcost]=inputsscwlp;
%N=10; [ukuran sampel]
%kapasitas=[39 35 31]; [kapasitas gudang lini 3]
%demand=[15 17 22 12]; [demand customer]
%suplai=[40 40]; [kapasitas gudang lini 2]
%jarak23=[120 150 175 200; 140 170 125 110; 170 180 125 115];
[jarak gudang lini 3 menuju lokasi customer]
%jarak13=[500 1250 1100 950; 1050 950 850 1100]; [jarak gudang
lini 2 menuju lokasi customer]
%biaya12=[50000 64000 73000; 44000 61000 49000]; [biaya pengiriman
per ton dari gudang lini 2 menuju gudang lini 3]
%biayasewa=[1500000;1250000;1000000]; [biaya sewa gudang lini 3]
%ULcost=[3000;2500;3300]; [biaya unloading per ton di setiap
gudang lini 3]

%initial parameter%
b=15;
n1=length(suplai);
n2=length(kapasitas);
n3=length(demand);
supply=[1:n1;suplai];
cap_gudang=[1:n2;kapasitas];
dem_cust=[1:n3;demand];
rho=0.3;
e=1e-14;
alpha=0.7;
maxiter=1000;
starttime=cputime;

%MEMBANGKITKAN SOLUSI AWAL%
for individu=1:N
d1(individu,:)=randperm(n1);
x1=d1;
d2(individu,:)=randperm(n2);
x2=d2;
d3(individu,:)=randperm(n3);
x3=d3;
end
n2_new=ceil(rand*n2);
x2=x2(:,1:n2_new);
for individu=1:N
```

```
urtn1((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=[ones(1,n1);x1(individu,:);supply(2,x1(individu,
:))];
urtn2((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=[ones(1,n2_new)*2;x2(individu,:);cap_gudang(2,x2
(individu,:))];
urtn3((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=[ones(1,n3)*3;x3(individu,:);dem_cust(2,x3(indiv
idu,:))];
urutan1=urtn1;
urutan2=urtn2;
urutan3=urtn3;
end
%struktur solusi gudang lini 2%
for individu=1:N
for j=1:n1
    if j==1
        urtn((((individu-1)*3)+1):(((individu-
1)*3)+3),j)=urutan1((((individu-1)*3)+1):(((individu-
1)*3)+3),1);
    else
        urtn((((individu-1)*3)+1):(((individu-1)*3)+3),((j-
1)*b)+1)=urutan1((((individu-1)*3)+1):(((individu-
1)*3)+3),j);
    end
end
urutan=urtn;
end
[~,panjangurutan1]=size(urutan);

%kasih space di akhir gudang lini 2 terakhir%
urutan(:,panjangurutan1+1:panjangurutan1+b-1)=zeros(3*N,b-
1);

%struktur solusi demand%
i2=ceil(n3/(b-1));
a=floor(n3/(b-1));
for individu=1:N
    for j=1:i2
        if j==1
            urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),2:b)=urutan3((((individu-1)*3)+1):(((individu-
1)*3)+3),1:(b-1));
        elseif j>a
            urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),(((j-1)*b)+2):(((j-1)*b)+2)-1+(n3-((b-1)*(j-
1))))=urutan3((((individu-1)*3)+1):(((individu-1)*3)+3),((b-
1)*(j-1))+1:n3);
        else
            urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),(((j-1)*b)+2):j*b)=urutan3((((individu-
1)*3)+1):(((individu-1)*3)+3),((b-1)*(j-1))+1:(b-1)*j);
        end
```

```
    end
end

%menyisipkan gudang lini 3%
[~,p]=size(urutan);
for individu=1:N
    urutan_ind=urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),:);
    urutan2_ind=urutan2((((individu-1)*3)+1):(((individu-
1)*3)+3),:);

    urutan_new=[1:p;urutan_ind];
    n=randperm(p);
    idx_lini3=n(:,1:n2_new);
    urutan2_new=[idx_lini3;urutan2_ind];
    urutan_new=[urutan_new,urutan2_new];
    x=urutan_new';
    y=sortrows(x);
    urutan_new=y';
    urutan_new=urutan_new(2:4,:);
    %menghilangkan kolom di matriks yang bernilai 0%
    nol=urutan_new(1,:)==0;
    urutan_new(:,nol)=[];
    urutan_pop((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=urutan_new;
end

%CEK KAPASITAS DAN ALOKASI%
[alokasi]=alokasicega(urutan_pop,N,x1,x3,n1,n2,n3,suplai);

%HITUNG FUNGSI TUJUAN
[biayatotal_pop]=hitungbiaya(N,n1,n2,n3,alokasi,biaya12,jara
k13,jarak23,biayasewa,ULcost);

%%%%HYBRID CEGA%%%%
%PEMILIHAN SAMPEL ELITE%
Pm=1;
A=2;
elite=quantile(biayatotal_pop,rho);
[ind_elite]=find(biayatotal_pop<=elite);
%PERHITUNGAN PARAMETER MUTASI%
ze=mean(biayatotal_pop(ind_elite,:));
z=min(biayatotal_pop);
u=ze/(2*z);
A_new=(1-alpha)*u+alpha*A;
Pm_new=A_new/2;
it=1;
while abs(Pm_new-Pm)>e && it<maxiter
    x1_new=x1;
    x2_new=x2;
    x3_new=x3;
    for individu=1:N
        urutan_elite=urutan_pop;
```

```
        %individu elite dipertahankan, non individu elite
dimutasi%
        if biayatotal_pop(individu)>elite
            %mutasi x1%
            batas=sort(ceil(n1*rand(1,2)));
            U=batas(1);
            B=batas(2);
            r=rand;
            if r<0.33
                x1(individu,U:B)=fliplr(x1_new(individu,U:B));
            elseif r<0.67
                x1(individu,[U B])=x1_new(individu, [U B]);
            else
                x1(individu,U:B)=x1_new(individu,[U+1:B U]);
            end
            %mutasi x2%
            x2_mt(individu,:)=randperm(n2);
            x2(individu,:)=x2_mt(individu,1:n2_new);
            %mutasi x3%
            batas=sort(ceil(n3*rand(1,2)));
            U=batas(1);
            B=batas(2);
            r=rand;
            if r<0.33
                x3(individu,U:B)=fliplr(x3_new(individu,U:B));
            elseif r<0.67
                x3(individu,[U B])=x3_new(individu,[U B]);
            else
                x3(individu,U:B)=x3_new(individu,[U+1:B U]);
            end
        end
    end
    %urutan setelah mutasi dan elitisme%
    for individu=1:N
    urtn1((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=[ones(1,n1);x1(individu,:);supply(2,x1(individu,
:))];
    urtn2((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=[ones(1,n2_new)*2;x2(individu,:);cap_gudang(2,x2
(individu,:))];
    urtn3((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=[ones(1,n3)*3;x3(individu,:);dem_cust(2,x3(indiv
idu,:))];
    urutan1=urtn1;
    urutan2=urtn2;
    urutan3=urtn3;
    end
    %struktur solusi gudang lini 2%
    for individu=1:N
    for j=1:n1
        if j==1
```

```
            urtn((((individu-1)*3)+1):(((individu-
1)*3)+3),j)=urutan1((((individu-1)*3)+1):(((individu-
1)*3)+3),1);
        else
            urtn((((individu-1)*3)+1):(((individu-
1)*3)+3),((j-1)*b)+1)=urutan1((((individu-
1)*3)+1):(((individu-1)*3)+3),j);
        end
    end
    urutan=urtn;
    end
    [~,panjangurutan1]=size(urutan);

    %kasih space di akhir gudang lini 2 terakhir%
    urutan(:,panjangurutan1+1:panjangurutan1+b-
1)=zeros(3*N,b-1);

    %struktur solusi demand%
    i2=ceil(n3/(b-1));
    a=floor(n3/(b-1));
    for individu=1:N
        for j=1:i2
            if j==1
                urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),2:b)=urutan3((((individu-1)*3)+1):(((individu-
1)*3)+3),1:(b-1));
            elseif j>a
                urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),(((j-1)*b)+2):(((j-1)*b)+2)-1+(n3-((b-1)*(j-
1))))=urutan3((((individu-1)*3)+1):(((individu-1)*3)+3),((b-
1)*(j-1))+1:n3);
            else
                urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),(((j-1)*b)+2):j*b)=urutan3((((individu-
1)*3)+1):(((individu-1)*3)+3),((b-1)*(j-1))+1:(b-1)*j);
            end
        end
    end

    %menyisipkan gudang lini 3%
    [~,p]=size(urutan);
    for individu=1:N
        urutan_ind=urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),:);
        urutan2_ind=urutan2((((individu-
1)*3)+1):(((individu-1)*3)+3),:);

        urutan_new=[1:p;urutan_ind];
        n=randperm(p);
        idx_lini3=n(:,1:n2_new);
        urutan2_new=[idx_lini3;urutan2_ind];
        urutan_new=[urutan_new,urutan2_new];
        x=urutan_new';
```

```
        y=sortrows(x);
        urutan_new=y';
        urutan_new=urutan_new(2:4,:);
        %menghilangkan kolom di matriks yang bernilai 0%
        nol=urutan_new(1,:)==0;
        urutan_new(:,nol)=[];
        urutan_pop((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=urutan_new;
    end

    %elitisme%
    for individu=ind_elite
        urutan_pop((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=urutan_elite((((individu-1)*3)+1):(((individu-
1)*3)+3),:);
    end

    urutan_pop=urutan_pop;

    %alokasi dan kapasitas%

[alokasi]=alokasicega(urutan_pop,N,x1,x3,n1,n2,n3,suplai);

    %fungsi tujuan%

[biayatotal_pop]=hitungbiaya(N,n1,n2,n3,alokasi,biaya12,jara
k13,jarak23,biayasewa,ULcost);

    %pemilihan sampel elite%
    elite=quantile(biayatotal_pop,rho);
    [ind_elite]=find(biayatotal_pop<=elite);

    %update parameter mutasi%
    Pm=Pm_new;
    A=A_new;
    ze=mean(biayatotal_pop(ind_elite,:));
    z=min(biayatotal_pop);
    u=ze/(2*z);
    A_new=(1-alpha)*u+alpha*A;
    Pm_new=A_new/2;
    it=it+1;
end
waktu=cputime-starttime
[mincost,idk]=min(biayatotal_pop);
alokasi_min=alokasi(((idk-1)*(n1+n2)+1):idk*(n1+n2),:);
```

Kode Program Algortima *Genetic Algorithm* untuk Menyelesaikan *Single Stage Capacitated Warehouse Location Problem*

```
function [mincost, alokasi_min]=GA_SSCWLP

%input%
```

```
[N, kapasitas, demand, suplai, jarak23, jarak13, biaya12,
biayasewa, ULcost]=inputsscwlp;

%initial parameter%
b=2;
n1=length(suplai);
n2=length(kapasitas);
n3=length(demand);
supply=[1:n1;suplai];
cap_gudang=[1:n2;kapasitas];
dem_cust=[1:n3;demand];
rho=0.1;
alpha=0.7;
maxiter=1000;
starttime=cputime;
Pm=0.3;
it=0;

    for individu=1:N
        d1(individu,:)=randperm(n1);
        x1=d1;
        d2(individu,:)=randperm(n2);
        x2=d2;
        d3(individu,:)=randperm(n3);
        x3=d3;
    end
    n2_new=ceil(rand*n2);
    x2=x2(:,1:n2_new);
    for individu=1:N
        urtn1((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=[ones(1,n1);x1(individu,:);supply(2,x1(individu,
:))];
        urtn2((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=[ones(1,n2_new)*2;x2(individu,:);cap_gudang(2,x2
(individu,:))];
        urtn3((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=[ones(1,n3)*3;x3(individu,:);dem_cust(2,x3(indiv
idu,:))];
        urutan1=urtn1;
        urutan2=urtn2;
        urutan3=urtn3;
        end
    %struktur solusi gudang lini 2%
    for individu=1:N
        for j=1:n1
            if j==1
                urtn((((individu-1)*3)+1):(((individu-
1)*3)+3),j)=urutan1((((individu-1)*3)+1):(((individu-
1)*3)+3),1);
            else
                urtn((((individu-1)*3)+1):(((individu-
1)*3)+3),((j-1)*b)+1)=urutan1((((individu-
1)*3)+1):(((individu-1)*3)+3),j);
```

```
            end
        end
        urutan=urtn;
    end
    [~,panjangurutan1]=size(urutan);

    %kasih space di akhir gudang lini 2 terakhir%
    urutan(:,panjangurutan1+1:panjangurutan1+b-
1)=zeros(3*N,b-1);

    %struktur solusi demand%
    i2=ceil(n3/(b-1));
    a=floor(n3/(b-1));
    for individu=1:N
        for j=1:i2
            if j==1
                urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),2:b)=urutan3((((individu-1)*3)+1):(((individu-
1)*3)+3),1:(b-1));
            elseif j>a
                urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),(((j-1)*b)+2):(((j-1)*b)+2)-1+(n3-((b-1)*(j-
1))))=urutan3((((individu-1)*3)+1):(((individu-1)*3)+3),((b-
1)*(j-1))+1:n3);
            else
                urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),(((j-1)*b)+2):j*b)=urutan3((((individu-
1)*3)+1):(((individu-1)*3)+3),((b-1)*(j-1))+1:(b-1)*j);
            end
        end
    end

    %menyisipkan gudang lini 3%
    [~,p]=size(urutan);
    for individu=1:N
        urutan_ind=urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),:);
        urutan2_ind=urutan2((((individu-
1)*3)+1):(((individu-1)*3)+3),:);

        urutan_new=[1:p;urutan_ind];
        n=randperm(p);
        idx_lini3=n(:,1:n2_new);
        urutan2_new=[idx_lini3;urutan2_ind];
        urutan_new=[urutan_new,urutan2_new];
        x=urutan_new';
        y=sortrows(x);
        urutan_new=y';
        urutan_new=urutan_new(2:4,:);
        %menghilangkan kolom di matriks yang bernilai 0%
        nol=urutan_new(1,:)==0;
        urutan_new(:,nol)=[];
```

```
        urutan_pop((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=urutan_new;
    end

    %CEK KAPASITAS DAN ALOKASI%

[alokasi]=alokasicega(urutan_pop,N,x1,x3,n1,n2,n3,suplai);

    %HITUNG FUNGSI TUJUAN

[biayatotal_pop]=hitungbiaya(N,n1,n2,n3,alokasi,biaya12,jara
k13,jarak23,biayasewa,ULcost);

while it<maxiter

    %PEMILIHAN SAMPEL ELITE%
    [~, idk_ind]=min(biayatotal_pop);

    %SALIN SAMPEL ELITE SEBANYAK 2 JIKA N GENAP%
    if mod(N,2)==0
        urutan_pop(1:3,:)=urutan_pop((((idk_ind-
1)*3)+1):(((idk_ind-1)*3)+3),:);
        urutan_pop(4:6,:)=urutan_pop((((idk_ind-
1)*3)+1):(((idk_ind-1)*3)+3),:);
        x1(1,:)=x1(idk_ind,:);
        x2(1,:)=x2(idk_ind,:);
        x3(1,:)=x3(idk_ind,:);
        x1(2,:)=x1(idk_ind,:);
        x2(2,:)=x2(idk_ind,:);
        x3(2,:)=x3(idk_ind,:);
    else
        urutan_pop(1:3,:)=urutan_pop((((idk_ind-
1)*3)+1):(((idk_ind-1)*3)+3),:);
        urutan_pop(4:6,:)=urutan_pop((((idk_ind-
1)*3)+1):(((idk_ind-1)*3)+3),:);
        urutan_pop(7:9,:)=urutan_pop((((idk_ind-
1)*3)+1):(((idk_ind-1)*3)+3),:);
        x1(1,:)=x1(idk_ind,:);
        x2(1,:)=x2(idk_ind,:);
        x3(1,:)=x3(idk_ind,:);
        x1(2,:)=x1(idk_ind,:);
        x2(2,:)=x2(idk_ind,:);
        x3(2,:)=x3(idk_ind,:);
        x1(3,:)=x1(idk_ind,:);
        x2(3,:)=x2(idk_ind,:);
        x3(3,:)=x3(idk_ind,:);
    end

    %MUTASI%
    x1_new=x1;
    x3_new=x3;
    urutan_elite=urutan_pop;
```

```
    for individu=1:N
        r1(individu)=rand;
        if r1(individu)<=Pm
            %mutasi x1%
            batas=sort(ceil(n1*rand(1,2)));
            U=batas(1);
            B=batas(2);
            r=rand;
            if r<0.33
                x1(individu,U:B)=fliplr(x1_new(individu,U:B));
            elseif r<0.67
                x1(individu,[U B])=x1_new(individu, [U B]);
            else
                x1(individu,U:B)=x1_new(individu,[U+1:B U]);
            end
            %mutasi x2%
            x2_mt(individu,:)=randperm(n2);
            x2(individu,:)=x2_mt(individu,1:n2_new);
            %mutasi x3%
            batas=sort(ceil(n3*rand(1,2)));
            U=batas(1);
            B=batas(2);
            r=rand;
            if r<0.33
                x3(individu,U:B)=fliplr(x3_new(individu,U:B));
            elseif r<0.67
                x3(individu,[U B])=x3_new(individu,[U B]);
            else
                x3(individu,U:B)=x3_new(individu,[U+1:B U]);
            end
        end
    end
        %urutan setelah mutasi dan elitisme%
    for individu=1:N
        urtn1((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=[ones(1,n1);x1(individu,:);supply(2,x1(individu,
:))];
        urtn2((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=[ones(1,n2_new)*2;x2(individu,:);cap_gudang(2,x2
(individu,:))];
        urtn3((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=[ones(1,n3)*3;x3(individu,:);dem_cust(2,x3(indiv
idu,:))];
        urutan1=urtn1;
        urutan2=urtn2;
        urutan3=urtn3;
    end
    %struktur solusi gudang lini 2%
    for individu=1:N
    for j=1:n1
        if j==1
```

```
            urtn((((individu-1)*3)+1):(((individu-
1)*3)+3),j)=urutan1((((individu-1)*3)+1):(((individu-
1)*3)+3),1);
        else
            urtn((((individu-1)*3)+1):(((individu-
1)*3)+3),((j-1)*b)+1)=urutan1((((individu-
1)*3)+1):(((individu-1)*3)+3),j);
        end
    end
    urutan=urtn;
    end
    [~,panjangurutan1]=size(urutan);

    %kasih space di akhir gudang lini 2 terakhir%
    urutan(:,panjangurutan1+1:panjangurutan1+b-
1)=zeros(3*N,b-1);

    %struktur solusi demand%
    i2=ceil(n3/(b-1));
    a=floor(n3/(b-1));
    for individu=1:N
        for j=1:i2
            if j==1
                urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),2:b)=urutan3((((individu-1)*3)+1):(((individu-
1)*3)+3),1:(b-1));
            elseif j>a
                urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),(((j-1)*b)+2):(((j-1)*b)+2)-1+(n3-((b-1)*(j-
1))))=urutan3((((individu-1)*3)+1):(((individu-1)*3)+3),((b-
1)*(j-1))+1:n3);
            else
                urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),(((j-1)*b)+2):j*b)=urutan3((((individu-
1)*3)+1):(((individu-1)*3)+3),((b-1)*(j-1))+1:(b-1)*j);
            end
        end
    end

    %menyisipkan gudang lini 3%
    [~,p]=size(urutan);
    for individu=1:N
        urutan_ind=urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),:);
        urutan2_ind=urutan2((((individu-
1)*3)+1):(((individu-1)*3)+3),:);

        urutan_new=[1:p;urutan_ind];
        n=randperm(p);
        idx_lini3=n(:,1:n2_new);
        urutan2_new=[idx_lini3;urutan2_ind];
        urutan_new=[urutan_new,urutan2_new];
        x=urutan_new';
```

```
        y=sortrows(x);
        urutan_new=y';
        urutan_new=urutan_new(2:4,:);
        %menghilangkan kolom di matriks yang bernilai 0%
        nol=urutan_new(1,:)==0;
        urutan_new(:,nol)=[];
        urutan_pop((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=urutan_new;
    end

    %elitisme%
    for individu=1:N
        if r1(individu)>Pm
            urutan_pop((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=urutan_elite((((individu-1)*3)+1):(((individu-
1)*3)+3),:);
        end
    end

    urutan_pop=urutan_pop;

    %alokasi dan kapasitas%

[alokasi]=alokasicega(urutan_pop,N,x1,x3,n1,n2,n3,suplai);

    %fungsi tujuan%

[biayatotal_pop]=hitungbiaya(N,n1,n2,n3,alokasi,biaya12,jara
k13,jarak23,biayasewa,ULcost);

    it=it+1;
end
waktu=cputime-starttime;
[mincost,idk]=min(biayatotal_pop);
alokasi_min=alokasi(((idk-1)*(n1+n2)+1):idk*(n1+n2),:);
```

Kode Program Algortima *Cross Entropy* untuk Menyelesaikan *Single Stage Capacitated Warehouse Location Problem*

```
function [mincost, alokasi_min]=CE_SSCWLP

%input%
[N, kapasitas, demand, suplai, jarak23, jarak13, biaya12,
biayasewa, ULcost]=inputsscwlp;

%initial parameter%
b=2;
n1=length(suplai);
n2=length(kapasitas);
n3=length(demand);
supply=[1:n1;suplai];
cap_gudang=[1:n2;kapasitas];
dem_cust=[1:n3;demand];
```

```
rho=0.1;
e=1e-10;
alpha=0.7;
maxiter=100;
starttime=cputime;
%Probabilitas Transisi%
P1_old=zeros(n1);
P2_old=zeros(n2);
P3_old=zeros(n3);
P1=ones(n1)*1/(n1-1);
P1=P1-diag(diag(P1));
P2=ones(n2)*1/(n2-1);
P2=P2-diag(diag(P2));
P3=ones(n3)*1/(n3-1);
P3=P3-diag(diag(P3));
kriteria=[max(max(abs(P1-P1_old))) max(max(abs(P2-P2_old)))
max(max(abs(P3-P3_old)))];
it=0;

while mean(kriteria)>e && it<maxiter
    for individu=1:N
        %membangkitkan x1%
        x1(individu,:)=rw(P1,n1);
        %membangkitkan x2%
        x2(individu,:)=rw(P2,n2);
        %membangkitkan x1%
        x3(individu,:)=rw(P3,n3);
    end
    for individu=1:N
        urtn1((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=[ones(1,n1);x1(individu,:);supply(2,x1(individu,
:))];
        urtn2((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=[ones(1,n2)*2;x2(individu,:);cap_gudang(2,x2(ind
ividu,:))];
        urtn3((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=[ones(1,n3)*3;x3(individu,:);dem_cust(2,x3(indiv
idu,:))];
        urutan1=urtn1;
        urutan2=urtn2;
        urutan3=urtn3;
        end
    %struktur solusi gudang lini 2%
    for individu=1:N
        for j=1:n1
            if j==1
                urtn((((individu-1)*3)+1):(((individu-
1)*3)+3),j)=urutan1((((individu-1)*3)+1):(((individu-
1)*3)+3),1);
            else
                urtn((((individu-1)*3)+1):(((individu-
1)*3)+3),((j-1)*b)+1)=urutan1((((individu-
1)*3)+1):(((individu-1)*3)+3),j);
```

```
            end
        end
        urutan=urtn;
    end
    [~,panjangurutan1]=size(urutan);

    %kasih space di akhir gudang lini 2 terakhir%
    urutan(:,panjangurutan1+1:panjangurutan1+b-
1)=zeros(3*N,b-1);

    %struktur solusi demand%
    i2=ceil(n3/(b-1));
    a=floor(n3/(b-1));
    for individu=1:N
        for j=1:i2
            if j==1
                urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),2:b)=urutan3((((individu-1)*3)+1):(((individu-
1)*3)+3),1:(b-1));
            elseif j>a
                urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),(((j-1)*b)+2):(((j-1)*b)+2)-1+(n3-((b-1)*(j-
1))))=urutan3((((individu-1)*3)+1):(((individu-1)*3)+3),((b-
1)*(j-1))+1:n3);
            else
                urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),(((j-1)*b)+2):j*b)=urutan3((((individu-
1)*3)+1):(((individu-1)*3)+3),((b-1)*(j-1))+1:(b-1)*j);
            end
        end
    end

    %menyisipkan gudang lini 3%
    [~,p]=size(urutan);
    for individu=1:N
        urutan_ind=urutan((((individu-1)*3)+1):(((individu-
1)*3)+3),:);
        urutan2_ind=urutan2((((individu-
1)*3)+1):(((individu-1)*3)+3),:);

        urutan_new=[1:p;urutan_ind];
        n=randperm(p);
        idx_lini3=n(:,1:n2);
        urutan2_new=[idx_lini3;urutan2_ind];
        urutan_new=[urutan_new,urutan2_new];
        x=urutan_new';
        y=sortrows(x);
        urutan_new=y';
        urutan_new=urutan_new(2:4,:);
        %menghilangkan kolom di matriks yang bernilai 0%
        nol=urutan_new(1,:)==0;
        urutan_new(:,nol)=[];
```

```
        urutan_pop((((individu-1)*3)+1):(((individu-
1)*3)+3),:)=urutan_new;
    end

    %CEK KAPASITAS DAN ALOKASI%

[alokasi]=alokasicega(urutan_pop,N,x1,x3,n1,n2,n3,suplai);

    %HITUNG FUNGSI TUJUAN

[biayatotal_pop]=hitungbiaya(N,n1,n2,n3,alokasi,biaya12,jara
k13,jarak23,biayasewa,ULcost);

    %PEMILIHAN SAMPEL ELITE%
    [biayatotal_pop, idk_ind]=sort(biayatotal_pop);
    jml_elite=floor(rho*N);

    %MATRIKS TRANSISI%
    w1=[];
    w2=[];
    w3=[];

    %UPDATE PROBABILITAS TRANSISI%
    %x1%
    tambahw1=zeros(n1,n1);
    x1_elt=x1(idk_ind,:);
    x1_elt=x1_elt(1:jml_elite,:);
    for i=1:jml_elite
        x1_updt=[];
        x1_updt=[x1_elt(i,:),x1_elt(i,1)];
        for j=1:n1
            baris=x1_updt(:,j);
            kolom=x1_updt(:,j+1);
            tambahw1(baris,kolom)=tambahw1(baris,kolom)+1;
        end
    end
    w1=tambahw1/jml_elite;
    P1=alpha.*w1+(1-alpha).*P1;

    %x2%
    tambahw2=zeros(n2,n2);
    x2_elt=x2(idk_ind,:);
    x2_elt=x2_elt(1:jml_elite,:);
    for i=1:jml_elite
        x2_updt=[];
        x2_updt=[x2_elt(i,:),x2_elt(i,1)];
        for j=1:n2
            baris=x2_updt(:,j);
            kolom=x2_updt(:,j+1);
            tambahw2(baris,kolom)=tambahw2(baris,kolom)+1;
        end
    end
    w2=tambahw2/jml_elite;
```

```
    P2=alpha.*w2+(1-alpha).*P2;

    %x3%
    tambahw3=zeros(n3,n3);
    x3_elt=x3(idk_ind,:);
    x3_elt=x3_elt(1:jml_elite,:);
    for i=1:jml_elite
        x3_updt=[];
        x3_updt=[x3_elt(i,:),x3_elt(i,1)];
        for j=1:n3
            baris=x3_updt(:,j);
            kolom=x3_updt(:,j+1);
            tambahw3(baris,kolom)=tambahw3(baris,kolom)+1;
        end
    end
    w3=tambahw3/jml_elite;
    P3=alpha.*w3+(1-alpha).*P3;

    it=it+1;

end
waktu=cputime-starttime
[mincost,idk]=min(biayatotal_pop);
alokasi_min=alokasi(((idk-1)*(n1+n2)+1):idk*(n1+n2),:);
```

Kode Program Pengecekan Kapasitas dan Alokasi untuk Algoritma *Hybrid* CE – GA, CE, dan GA

```
function
[alokasi]=alokasicega(urutan_pop,N,x1,x3,n1,n2,n3,suplai)

%CEK KAPASITAS DAN ALOKASI%
%hapus gudang lini 2%
[~,w]=size(urutan_pop);
alokasi=zeros(N*(n1+n2),n2+n3);

for individu=1:N
    urutan=urutan_pop(((individu-1)*3+1):individu*3,:);
    g1=x1(individu,:);
    g3=x3(individu,:);
    urtn23((((individu-1)*4)+1):(((individu)*4)),:)=[urutan;
1:w];
    urutan23=urtn23((((individu-
1)*4)+1):(((individu)*4)),:);
    gudang1=urutan23(1,:)==1;
    urutan23(:,gudang1)=[];

    %hapus gudang lini 3 yang tidak dibuka%
    ff=find(urutan23(1,:)==3,1);
    [~,h]=size(urutan23);
    if isempty(ff)==0
    ff2=find(urutan23(1,:)==2);
    for i=1:length(ff2)
```

```
        if i==length(ff2)
        if ff2(:,i)==h
        urutan23(:,ff2(:,i))=0;
        else
        urutan23(:,ff2(:,i))=urutan23(:,ff2(:,i));
        end
        else
        if ff2(:,i)==ff2(:,i+1)-1
        urutan23(:,ff2(:,i))=0;
        end
        end
        end
        hapusnol=urutan23(1,:)==0;
        urutan23(:,hapusnol)=[];
        else
        urutan23=0;
        hps=urutan23==0;
        urutan23(:,hps)=[];
        end
        %cek kapasitas gudang 2%

        if isempty(urutan23)==0
        cc=find(urutan23(1,:)==2);
        urutany=urutan;
        [~,h]=size(urutan23);
        for ii=1:length(cc)
        if ii==length(cc)
        suplaidemand=urutan23(3,cc(:,ii)+1:h);
        indekssupdem=urutan23(4,cc(:,ii)+1:h);
        else
        suplaidemand=urutan23(3,cc(:,ii)+1:(cc(:,ii+1)-1));
        indekssupdem=urutan23(4,cc(:,ii)+1:(cc(:,ii+1)-1));
        end
        suplaigudang2=0;
        for iii=1:length(suplaidemand)
        if sum([suplaigudang2
    suplaidemand(:,iii)])>urutan23(3,cc(:,ii))
        I=indekssupdem(:,iii);
        satudua=find(urutan(1,:)<3);
        batasj=find(satudua>I);
        if isempty(batasj)==1
        J=2;
        urutan(:,J:I)=urutany(:,[I J:I-1]);
        else
        J=satudua(:,batasj(:,1));
        urutan(:,I:J)=urutany(:,[J I:J-1]);
        end
        break
        else
        suplaigudang2=[suplaigudang2 suplaidemand(:,iii)];
        end
        end
        end
```

```
    end

    %hapus gudang lini 3 yang tidak dibuka%
    urutan123=urutan;
    cekgudang2=find(urutan123(1,:)==2);
    for nn=1:length(cekgudang2)
        if nn==length(cekgudang2)
            if cekgudang2(:,nn)==length(urutan123)
                urutan123(:,cekgudang2(:,nn))=0;
            elseif urutan123(1,cekgudang2(:,nn)+1)==1
                urutan123(:,cekgudang2(:,nn))=0;
            else

urutan123(:,cekgudang2(:,nn))=urutan123(:,cekgudang2(:,nn));
            end
        else
            if cekgudang2(:,nn)==cekgudang2(:,nn+1)-1
                urutan123(:,cekgudang2(:,nn))=0;
            end
        end
    end
    hapusnol=urutan123(1,:)==0;
    urutan123(:,hapusnol)=[];

    %kebutuhan suplai dari gudang lini 3%
    cd=find(urutan123(1,:)==2); %gudang penyangga/gudang
lini 3%
    f1=find(urutan123(1,:)<3); %gudang lini 2 dan gudang
lini 3%
    demand23=urutan123; %urutan yang sudah menghilangkan
gudang lini 3 tidak terpakai%
    alokasi23=zeros(n2,n3); %bikin matriks kosong untuk
alokasi lini 3 ke demand%
    find2=find(demand23(1,:)==2,1); %gudang lini 3 yang
paling depan%
    if isempty(find2)==0 %dilakukan jika ada gudang lini 3
yang dibuka%
        suplaigudang1=[];
        for mm=1:length(cd)
            f2=find(f1>cd(:,mm)); %gudang lini 2 dan lini3
yang terletak setelah gudang lini 3 ke-mm%
            if isempty(f2)==1; %kalo gudang di paling
belakang%

sg1=urutan123(3,cd(:,mm)+1:length(urutan123)); %supplai
gudang lini 3 = demand yang dibelakangnya%
                demand23(:,cd(:,mm)+1:length(urutan123))=0;
%yang sudah dialokasikan di 0 kan%

indeks2=urutan123(2,cd(:,mm)+1:length(urutan123)); %indeks
lokasi demand%
            else
```

```
                    sg1=urutan123(3,cd(:,mm)+1:f1(:,f2(:,1))-1);
%nilai supplai%
                    demand23(:,cd(:,mm)+1:f1(:,f2(:,1))-1)=0;
%meng-0-kan%

indeks2=urutan123(2,cd(:,mm)+1:f1(:,f2(:,1))-1); %indeks
lokasi demand%
                end
                suplaigudang1(:,mm)=sum(sg1);

                %standardisasi alokasi gudang lini 3 ke demand%
                for nn=1:length(sg1)

alokasi23(urutan123(2,cd(:,mm)),indeks2(:,nn))=sg1(:,nn);
                end
            end
            hapusnol=demand23(1,:)==0;
            demand23(:,hapusnol)=[];

            ce=demand23(1,:)==2;
            demand23(3,ce)=suplaigudang1;
        end

        %alokasi dari gudang lini 2%
        hapus1=demand23(1,:)==1;
        demand23(:,hapus1)=[];
        [~, k]=size(demand23);
        alokasi1=zeros(n1,k);
        for i=1:n1
            for j=1:k
                if sum(alokasi1(i,:),2)>=suplai(:,g1(:,i))
                    break
                end
                if sum(alokasi1(:,j),1)==demand23(3,j)
                    alokasi1(i,j)=0;
                else
                    if demand23(1,j)==3
                        if suplai(:,g1(:,i))-
sum(alokasi1(i,1:j),2)<demand23(3,j)
                            alokasi1(i,j)=0;
                            break
                        elseif suplai(:,g1(:,i))-
sum(alokasi1(i,1:j),2)>=demand23(3,j)
                            alokasi1(i,j)=demand23(3,j);
                        elseif suplai(:,g1(:,i))-
sum(alokasi1(i,1:j),2)<=0
                            alokasi1(i,j)=0;
                        end
                    else
                        if suplai(:,g1(:,i))-
sum(alokasi1(i,1:j),2)<demand23(3,j)
                            alokasi1(i,j)=suplai(:,g1(:,i))-
sum(alokasi1(i,1:j),2);
```

```
                    elseif suplai(:,g1(:,i))-
sum(alokasi1(i,1:j),2)>=demand23(3,j)
                        alokasi1(i,j)=demand23(3,j)-
sum(alokasi1(:,j),1);
                    elseif suplai(:,g1(:,i))-
sum(alokasi1(i,1:j),2)<=0
                        alokasi1(i,j)=0;
                    end
                end
            end
        end

    %standardisasi alokasi gudang lini 2 ke gudang lini 3
dan demand%
    alokasi12=zeros(n1,n2);
    alokasi13=zeros(n1,n3);
    indeks=demand23(2,:);
    [~, k]=size(demand23);
    for ii=1:n1
        for jj=1:k
            if demand23(1,jj)==2
                alokasi12(ii,indeks(:,jj))=alokasi1(ii,jj);
            else
                alokasi13(ii,indeks(:,jj))=alokasi1(ii,jj);
            end
        end
    end
    [~, r]=sort(g1);
    rt=r';
    alokasi12urut=alokasi12(rt,:);
    alokasi13urut=alokasi13(rt,:);
    alokasi(((individu-
1)*(n1+n2)+1):individu*(n1+n2),:)=[alokasi12urut
alokasi13urut; zeros(n2,n2) alokasi23];
end
```

Kode Program Perhitungan Biaya untuk Algortima *Hybrid* CE – GA, CE, dan GA

```
function
[biayatotal_pop]=hitungbiaya(N,n1,n2,n3,alokasi,biaya12,jara
k13,jarak23,biayasewa,ULcost)
 for individu=1:N
     alokasi_ind=alokasi(((individu-
1)*(n1+n2)+1):individu*(n1+n2),:);
     alokasi12=alokasi_ind(1:n1,1:n2);
     alokasi13=alokasi_ind(1:n1,n2+1:n2+n3);
     alokasi23=alokasi_ind(n1+1:n1+n2,n2+1:n2+n3);
     biayatrans12=alokasi12.*biaya12;
     biayatrans13=alokasi13.*biaya(jarak13);
     biayatrans23=alokasi23.*biaya(jarak23);
```

```
biayatransportasi=sum(sum(biayatrans12))+sum(sum(biayatrans1
3))+sum(sum(biayatrans23));
      %biaya sewa warehouse%
      keluar_lini2=sum(alokasi23,2);
      biner=keluar_lini2>0;
      sewa_lini2=biner.*biayasewa';
      biayasewa_lini2=sum(sewa_lini2);
      %biaya unloading%
      biaya_unloading=keluar_lini2.*ULcost';
      ULcost_lini2=sum(biaya_unloading);
      %biaya total%

biayatotal(individu,:)=biayatransportasi+biayasewa_lini2+ULc
ost_lini2;
 end
 biayatotal_pop=biayatotal;
```

LAMPIRAN 4
HASIL KOMPUTASI METODE EKSAK

ALOKASI	JUMLAH (TON)
GP Lhokseumawe	**590**
Aceh Singkil Utara	245
Labuan Batu Selatan	250
Kota Langsa	45
Aceh Timur	50
GP Medan	**4267**
Toba Samosir	38
Labuan Batu Utara	129
Serdang bedagai	580
Kota Binjai	40
Kota Medan	19
Kota Tebing Tinggi	25
Kota Pematang Siantar	175
Kab. Batu Bara	200
Samosir	125
Pakpak Bharat	100
Humbang Hasundutan	100
Langkat	125
Deli Serdang	250
Karo	240
Dairi	250
Simalungun	460
Kab. Asahan	225
Tapanuli Utara	191
Tapanuli Tengah	118
Tapanuli Selatan	150
Nias	20
Kota Subulussalam	397
Aceh Tamiang	65
Aceh Singkil	245
DC Padang	**7996**
Kab. Dharmasraya	465
Sungai Penuh	130
Bungo	344
Tebo	481
Kerinci	557
Kab. Indragiri Hulu	555
Kuantan Singingi	347
Kota Pariaman	100
Kota Payakumbuh	100
Kota Bukittinggi	15

ALOKASI	JUMLAH (TON)
Kota Padang Panjang	15
Kota Sawah Lunto	28
Kota Solok	30
Kota Padang	332
Pasaman barat	855
Kab. Solok Selatan	1033
Pasaman	264
Lima Puluh Kota	333
Agam	580
Padang Pariaman	300
Tanah Datar	428
Sijunjung	120
Solok	30
Pesisir Selatan	454
Mandailing Natal	100
GP Dumai	**2030**
Kota Dumai	47
Bintan	20
Kep. Meranti	110
Kota Pekan Baru	32
Rokan Hilir	216
Bengkalis	337
Rokan Hulu	269
Kampar	390
Pelalawan	430
Kota Padang Sidempua	75
Labuhan Batu	104
GP Jambi	**530**
Kota Jambi	16
Kab. Batang Hari	256
Kab. Indragiri Hilir	258
GP Palembang	**7516**
Ogan Komering Ulu Timur	336
Kab Ogan Ilir	344
Mesuji	520
Tulang Bawang	994
Kaur	370
Kab. Panukal. Abab Lem. Ilir	76
Kota Palembang	24
Banyuasin	460
Musi Banyuasin	878
Muara Enim	231
Ogan Komering Ilir	645
Ogan Komering Ulu	232
Tanjung Jabung Barat	384
Tanjung Jabung Timur	536

ALOKASI	JUMLAH (TON)
Muaro Jambi	256
Sarolangun	314
GP Merangin	***916***
Merangin	916
GP Bengkulu	**4702**
Kab. Kepahiang	66
Kab Empat Lawang	185
Kota Bengkulu	48
Bengkulu Tengah	235
Lebong	242
Mukomuko	1133
Seluma	272
Kab. Bengkulu Utara	849
Rejang Lebong	399
Bengkulu Selatan	341
Musi Rawas Utara	192
Kota Lubuk Linggau	64
Kota Pagar Alam	126
Musi Rawas	404
Lahat	146
DC Lampung	**6892**
Kab Pesawaran	724
Pesisir Barat	428
Kota Metro	250
Kota Bandar Lampung	88
Tulang Bawang Barat	240
Pringsewu	340
Way Kanan	700
Lampung Timur	1576
Lampung Selatan	1142
Kab. Tanggamus	905
Lampung Barat	477
Kota Prabumulih	22
GP Tanjung Pinang	**3267**
Kota Tanjung Pinang	12
Kota Batam	7
Karimun Tanung	8
Belitung	110
Lampung Tengah	1276
GP Kotabumi	***1854***
Ogan Komering Ulu Selatan	458
Gunung Sitoli	29
Lampung Utara	436
Padang Lawas Utara	931
GP Pangkal Pinang	**1158**
Kota Pangkal Pinang	13

ALOKASI	JUMLAH (TON)
Belitung Timur	150
Kab. Bangka Barat	245
Bangka Tengah	284
Bangka Selatan	196
Bangka	270

KETERANGAN:
Gudang Lini 2
Gudang Lini 3
Customer

LAMPIRAN 5
HASIL KOMPUTASI ALGORITMA *HYBRID* CE – GA

ALOKASI	JUMLAH (TON)
GP Lhokseumawe	**540**
Aceh Singkil Utara	245
Labuan Batu Selatan	250
Kota Langsa	45
GP Medan	**3389**
Toba Samosir	38
Labuan Batu Utara	129
Serdang bedagai	580
Kota Binjai	40
Kota Medan	19
Kota Tebing Tinggi	25
Kota Pematang Siantar	175
Kab. Batu Bara	200
Samosir	125
Humbang Hasundutan	100
Deli Serdang	250
Karo	240
Dairi	250
Simalungun	460
Tapanuli Utara	191
Tapanuli Selatan	150
Nias	20
Kota Subulussalam	397
DC Padang	**7017**
Kab. Dharmasraya	465
Sungai Penuh	130
Bungo	344
Tebo	481
Kab. Indragiri Hulu	555
Kuantan Singingi	347
Kota Sawah Lunto	28
Kota Padang	332
Pasaman barat	855
Pasaman	264
Lima Puluh Kota	333
Agam	580
Padang Pariaman	300
Tanah Datar	428

ALOKASI	JUMLAH (TON)
Sijunjung	120
Pesisir Selatan	454
Langkat	125
Mandailing Natal	100
GP Bukittinggi	***776***
Ogan Komering Ulu Timur	*336*
Kota Payakumbuh	*100*
Kota Bukittinggi	*15*
Kota Padang Panjang	*15*
Aceh Tamiang	*65*
Aceh Singkil	*245*
GP Dumai	**2971**
Kota Dumai	47
Bintan	20
Kep. Meranti	110
Rokan Hilir	216
Rokan Hulu	269
Pelalawan	430
Kota Padang Sidempua	75
Labuhan Batu	104
Aceh Timur	50
GP Solok	***1650***
Kerinci	557
Solok	30
Kab Solok Selatan	1033
Kota Solok	30
GP Jambi	**855**
Kota Jambi	16
Kab. Batang Hari	256
Kab. Indragiri Hilir	258
Pakpak Bharat	100
Kab. Asahan	225
GP Palembang	**7570**
Kab Ogan Ilir	344
Mesuji	520
Tulang Bawang	994
Kaur	370
Kab. Panukal. Abab Lem. Ilir	76
Kota Palembang	24
Banyuasin	460
Musi Banyuasin	878
Muara Enim	231

ALOKASI	JUMLAH (TON)
Ogan Komering Ilir	645
Ogan Komering Ulu	232
Tanjung Jabung Barat	384
Tanjung Jabung Timur	536
Muaro Jambi	256
Sarolangun	314
Kampar	390
GP Merangin	***916***
Merangin	916
GP Bengkulu	**4820**
Kab. Kepahiang	66
Kab Empat Lawang	185
Kota Bengkulu	48
Bengkulu Tengah	235
Lebong	242
Mukomuko	1133
Seluma	272
Kab. Bengkulu Utara	849
Rejang Lebong	399
Bengkulu Selatan	341
Musi Rawas Utara	192
Kota Lubuk Linggau	64
Kota Pagar Alam	126
Musi Rawas	404
Lahat	146
Tapanuli Tengah	118
DC Lampung	**6924**
Kab Pesawaran	724
Pesisir Barat	428
Kota Metro	250
Kota Bandar Lampung	88
Tulang Bawang Barat	240
Pringsewu	340
Way Kanan	700
Lampung Timur	1576
Lampung Selatan	1142
Kab. Tanggamus	905
Lampung Barat	477
Kota Prabumulih	22
Kota Pekan Baru	32
GP Tanjung Pinang	**3267**
Kota Tanjung Pinang	12
Kota Batam	7

ALOKASI	JUMLAH (TON)
Karimun Tanung	8
Belitung	110
Lampung Tengah	1276
GP Kotabumi	***1854***
Ogan Komering Ulu Selatan	458
Gunung Sitoli	29
Lampung Utara	436
Padang Lawas Utara	931
GP Pangkal Pinang	**1258**
Kota Pangkal Pinang	13
Belitung Timur	150
Kab. Bangka Barat	245
Bangka Tengah	284
Bangka Selatan	196
Bangka	270
Kota Pariaman	100

KETERANGAN:
Gudang Lini 2
Gudang Lini 3
Customer

BIOGRAFI PENULIS

Fatmah Munif Lahdji, yang merupakan anak bungsu dari empat bersaudara, lahir di Surabaya pada 14 Desember 1994. Penulis menempuh pendidikan formal dimulai dari TK Alirsyad Surabaya (1999-2000), SD Muhammadiyah 4 Surabaya (2000-2006), SMP Alhikmah Surabaya (2006-2009), dan SMA Alhikmah Surabaya (2009-2012). Penulis menyelesaikan pendidikan S1 pada tahun 2016 di Jurusan Teknik Industri, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember, Surabaya. Selama masa perkuliahan penulis aktif dalam beberapa kegiatan kepanitian yang dilaksanakan oleh organisasi mahasiswa, seperti HMTI ITS dan BEM FTI ITS. Selain itu, penulis juga aktif mengikuti kegiatan seperti lomba keilmuan. Penulis pernah meraih posisi kedua dalam lomba keilmuan Teknik Industri tingkat nasional (ISMEC) yang diadakan oleh Universitas Brawijaya pada tahun 2015. Penulis dapat dihubungi melalui fatmahlahdji@gmail.com